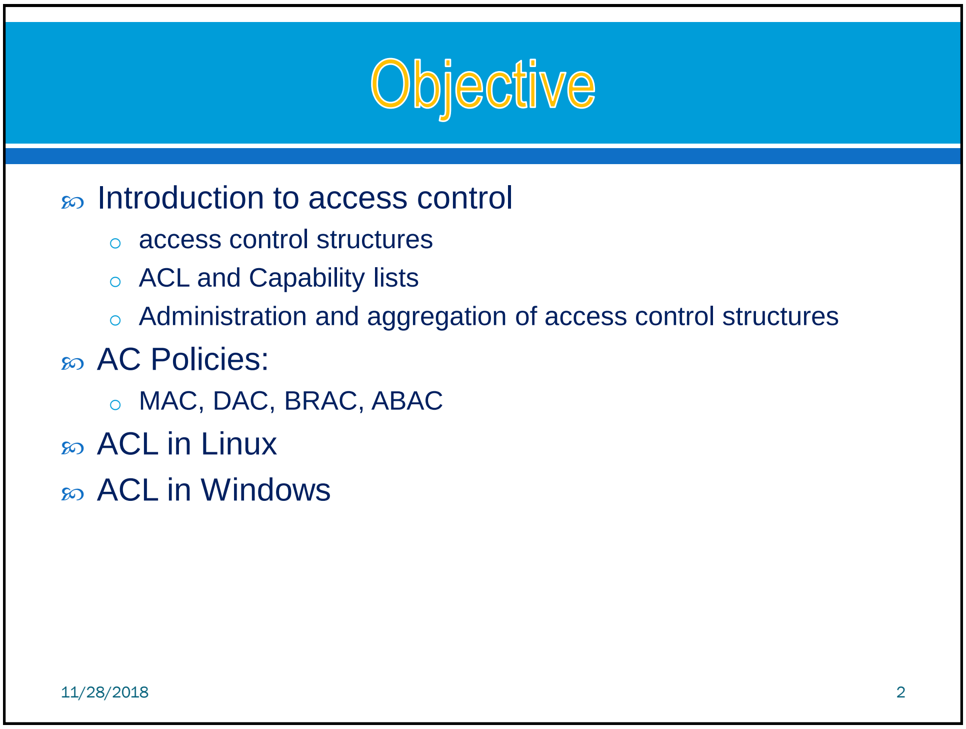
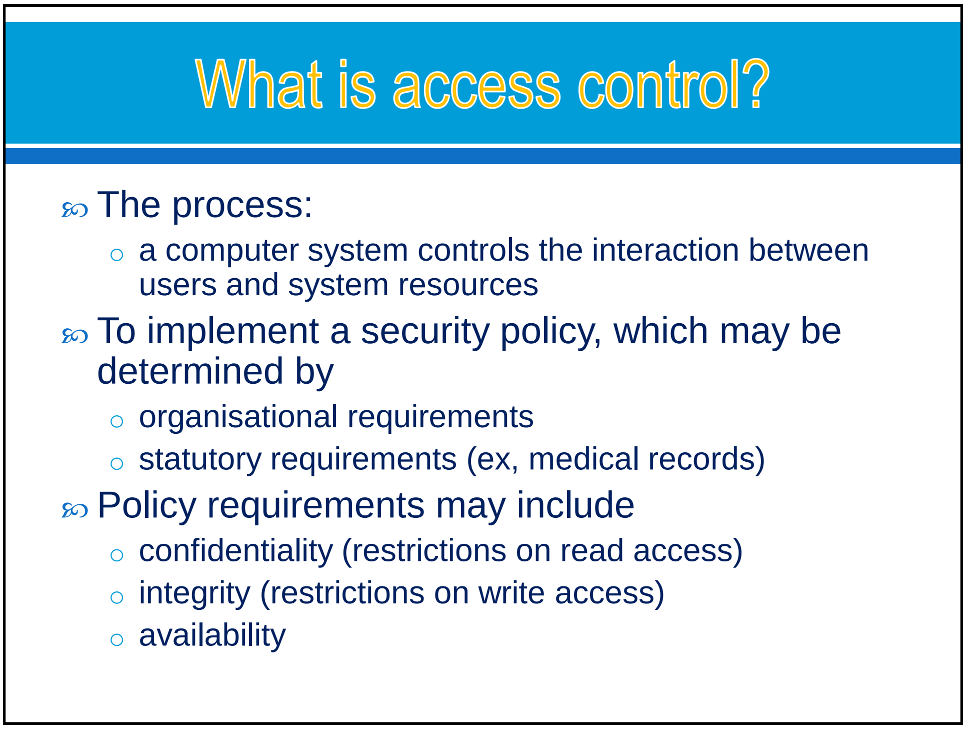
Access control

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE



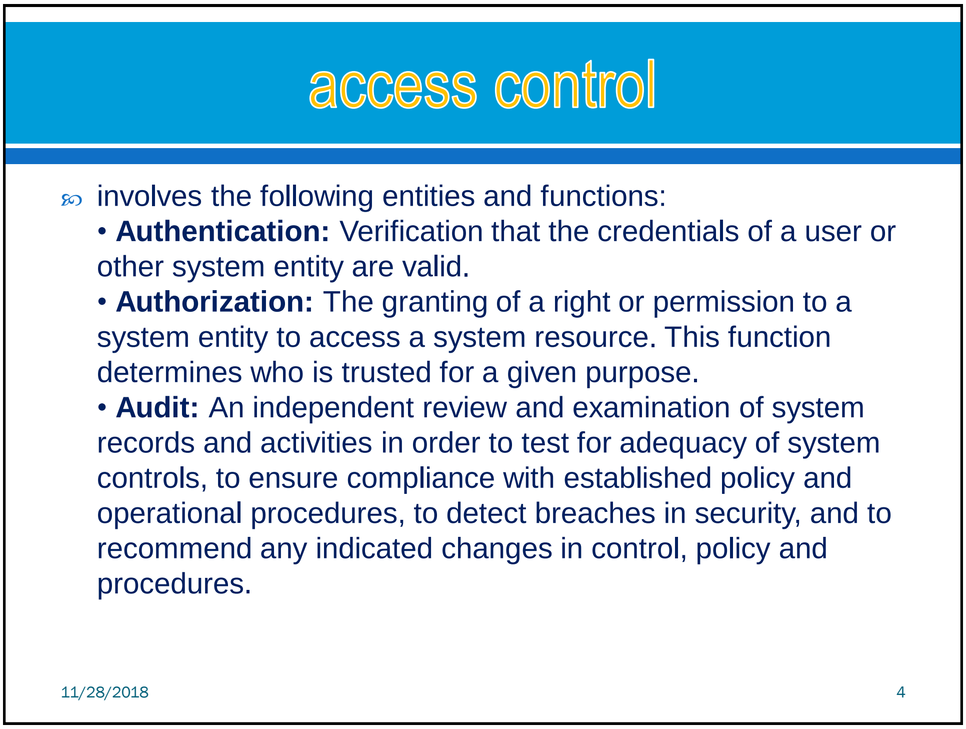
Objective

- Introduction to access control
  - access control structures
  - ACL and Capability lists
  - Administration and aggregation of access control structures
- AC Policies:
  - MAC, DAC, BRAC, ABAC
- ACL in Linux
- ACL in Windows



What is access control?

- The process:
  - a computer system controls the interaction between users and system resources
- To implement a security policy, which may be determined by
  - organisational requirements
  - statutory requirements (ex, medical records)
- Policy requirements may include
  - confidentiality (restrictions on read access)
  - integrity (restrictions on write access)
  - availability



access control

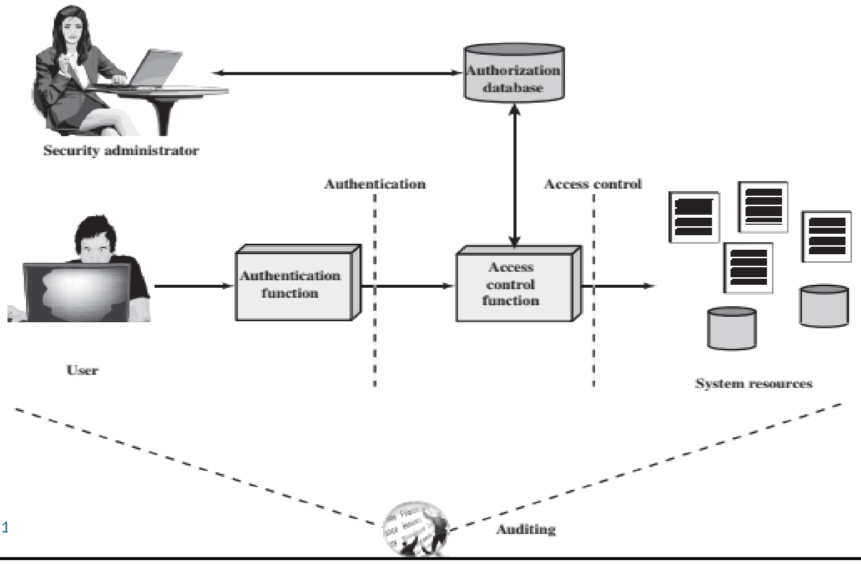
- involves the following entities and functions:

  • **Authentication:** Verification that the credentials of a user or other system entity are valid.

  • **Authorization:** The granting of a right or permission to a system entity to access a system resource. This function determines who is trusted for a given purpose.

  • **Audit:** An independent review and examination of system records and activities in order to test for adequacy of system controls, to ensure compliance with established policy and operational procedures, to detect breaches in security, and to recommend any indicated changes in control, policy and procedures.

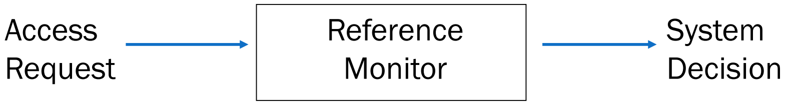
Relationship Among Access Control and Other Security Functions



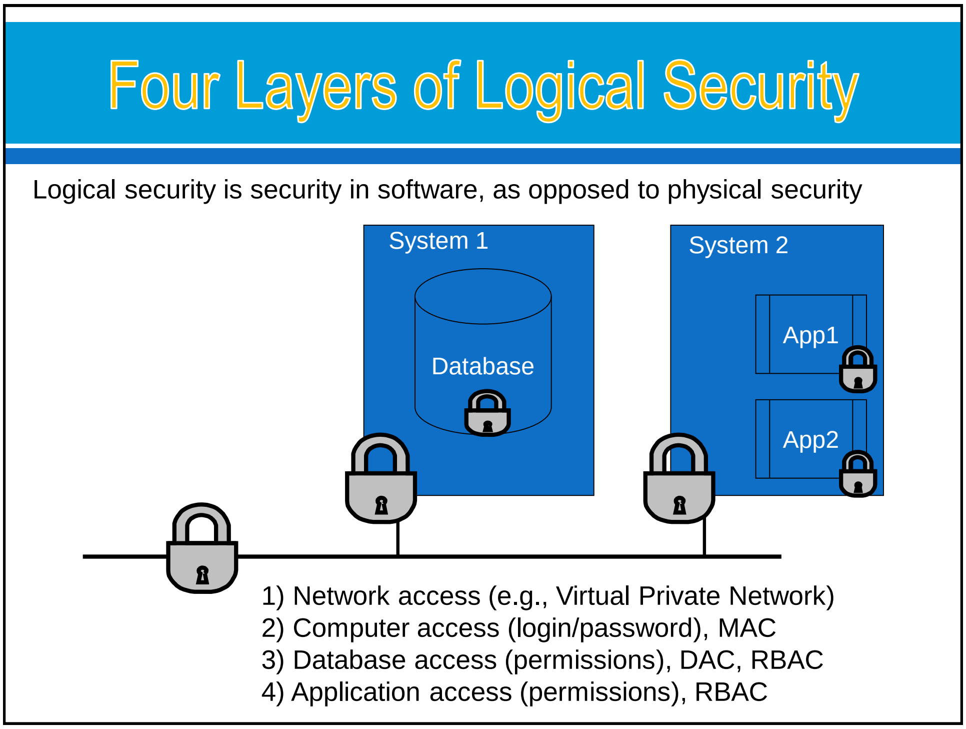
1

A schematic view

- A user requests access (read, write, print, etc.) to a resource in the computer system
- The *reference monitor*
  - establishes the validity of the request …
  - … and returns a decision either granting or denying access to the user



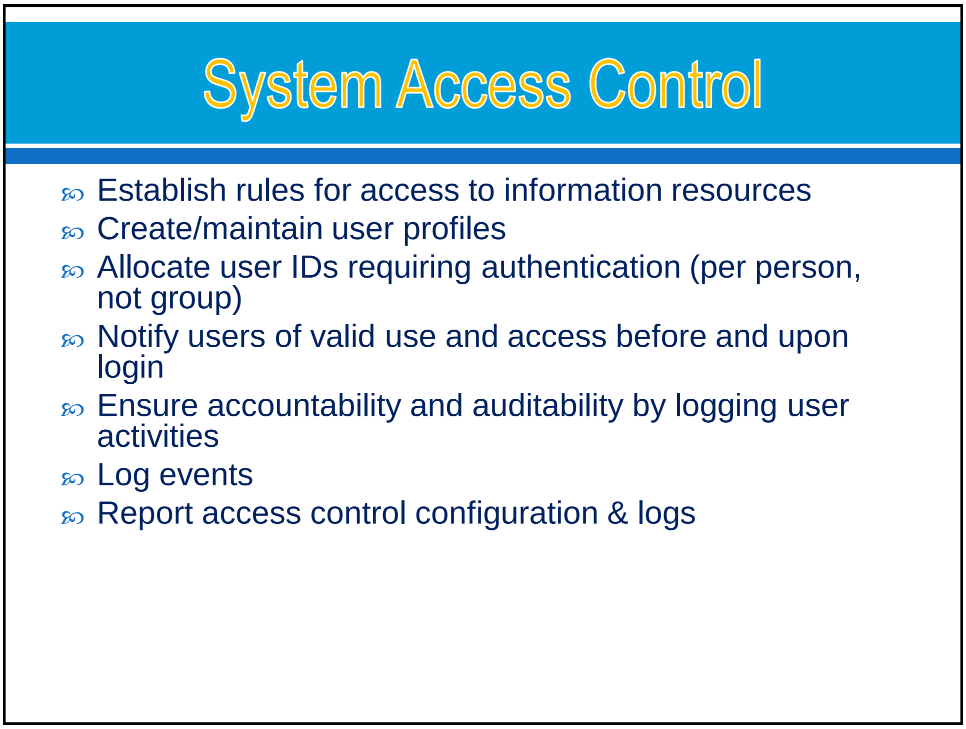
- Ex: RM
  - a paper-based office: the set of (locked) filing cabinets
  - a night club: the security guard + the guest list



Four Layers of Logical Security

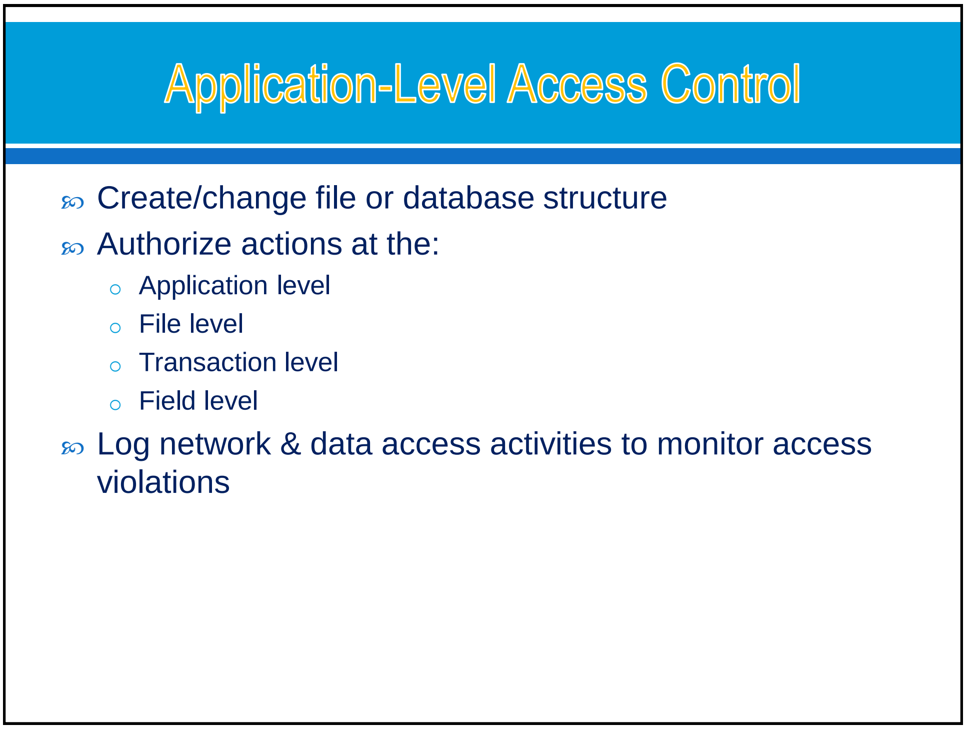
Logical security is security in software, as opposed to physical security

1) Network access (e.g., Virtual Private Network)
2) Computer access (login/password), MAC
3) Database access (permissions), DAC, RBAC
4) Application access (permissions), RBAC



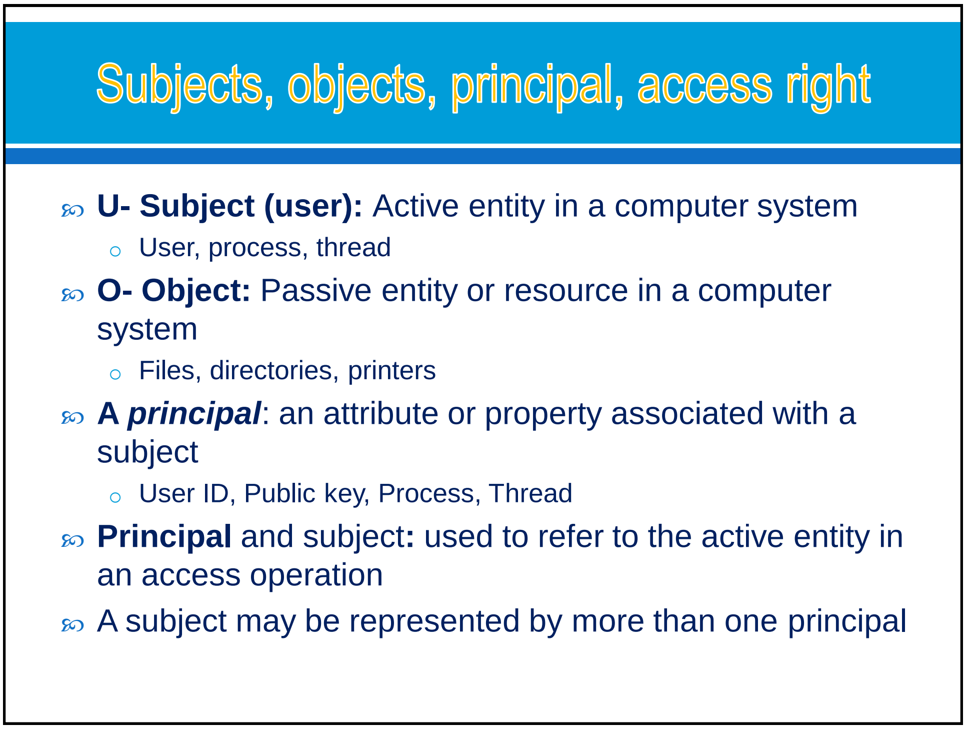
System Access Control

- Establish rules for access to information resources
- Create/maintain user profiles
- Allocate user IDs requiring authentication (per person, not group)
- Notify users of valid use and access before and upon login
- Ensure accountability and auditability by logging user activities
- Log events
- Report access control configuration & logs



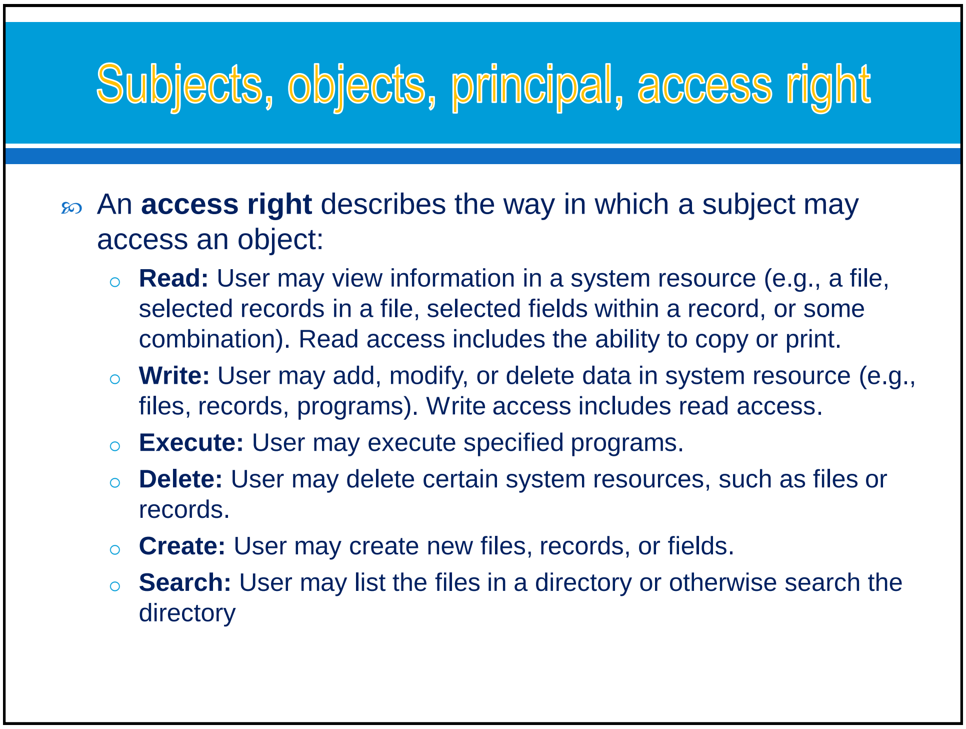
Application-Level Access Control

- Create/change file or database structure
- Authorize actions at the:
  - Application level
  - File level
  - Transaction level
  - Field level
- Log network & data access activities to monitor access violations



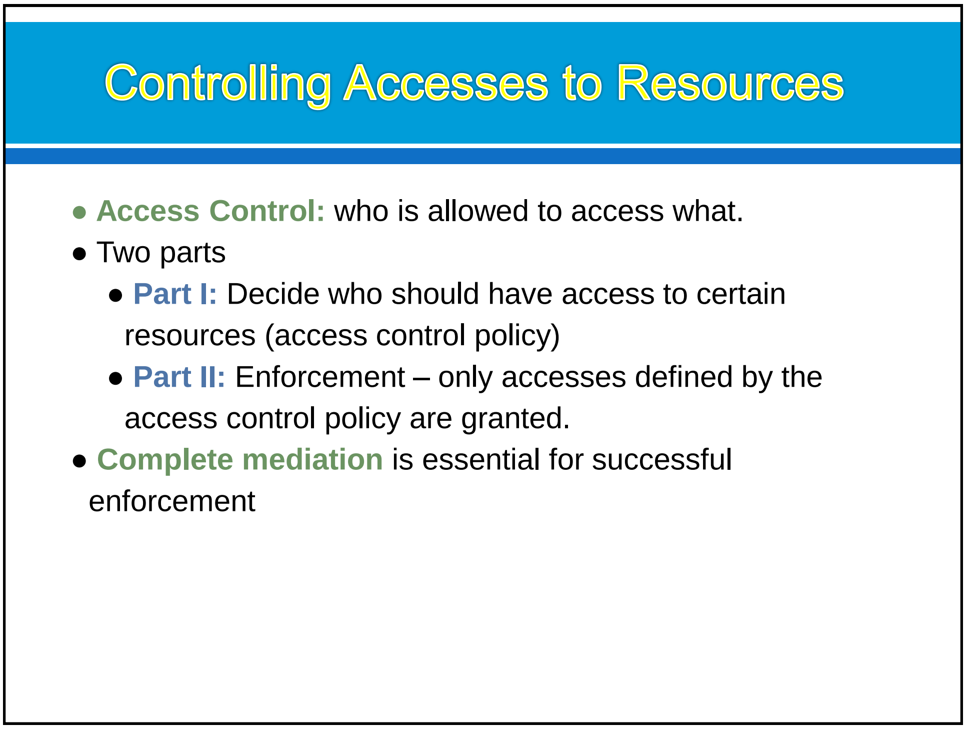
Subjects, objects, principal, access right

- **U- Subject (user):** Active entity in a computer system
  - User, process, thread
- **O- Object:** Passive entity or resource in a computer system
  - Files, directories, printers
- **A *principal***: an attribute or property associated with a subject
  - User ID, Public key, Process, Thread
- **Principal** and subject**:** used to refer to the active entity in an access operation
- A subject may be represented by more than one principal



Subjects, objects, principal, access right

- An **access right** describes the way in which a subject may access an object:
  - **Read:** User may view information in a system resource (e.g., a file, selected records in a file, selected fields within a record, or some combination). Read access includes the ability to copy or print.
  - **Write:** User may add, modify, or delete data in system resource (e.g., files, records, programs). Write access includes read access.
  - **Execute:** User may execute specified programs.
  - **Delete:** User may delete certain system resources, such as files or records.
  - **Create:** User may create new files, records, or fields.
  - **Search:** User may list the files in a directory or otherwise search the directory



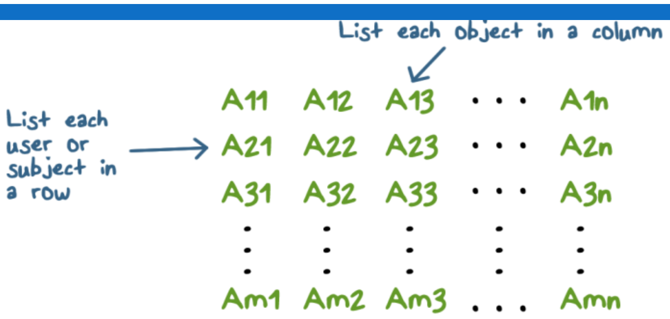
Controlling Accesses to Resources

- **Access Control:** who is allowed to access what.
- Two parts
  - **Part I:** Decide who should have access to certain resources (access control policy)
  - **Part II:** Enforcement – only accesses defined by the access control policy are granted.
- **Complete mediation** is essential for successful enforcement

Access Control Matrix (ACM)

- Introduced by Lampson (1972) and extended by Harrison, Ruzzo and Ullman (1976-8)
- An access control matrix (ACM) **abstracts the state relevant to access control.**
- Rows of ACM correspond to users/subjects/groups
- Columns correspond to resources that need to be protected.
- ACM defines who can access what
  - ACM [U,O] define what access rights user U has for object O.

The access control matrix

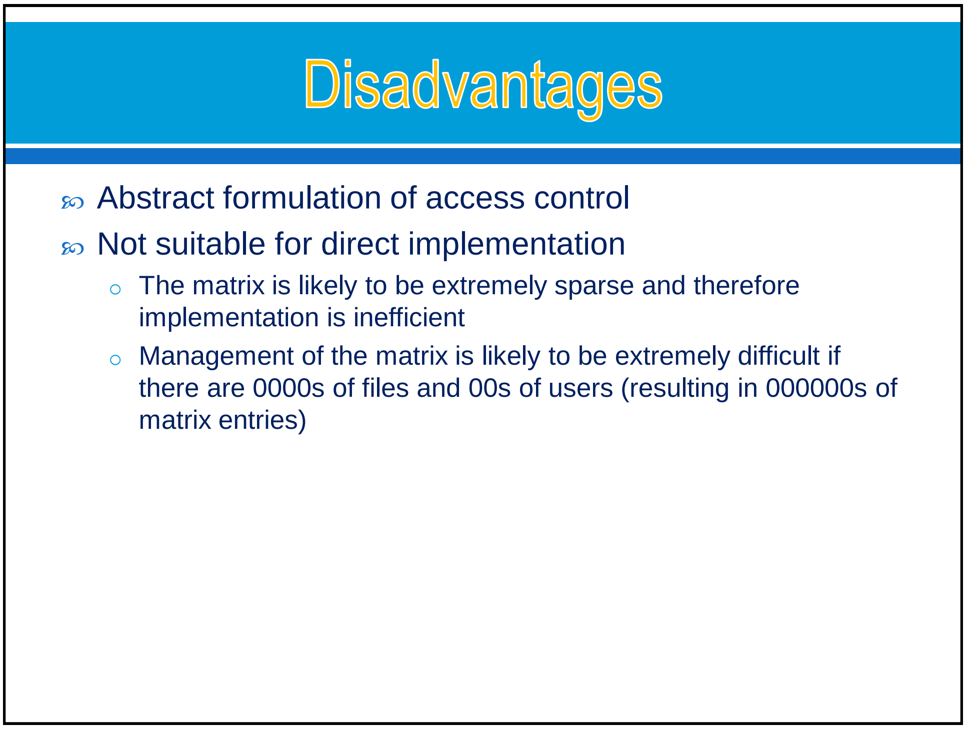


List each object in a column
List each user or subject in a row

$$
\begin{array}{ccccc}
A11 & A12 & A13 & \cdots & A1n \\
A21 & A22 & A23 & \cdots & A2n \\
A31 & A32 & A33 & \cdots & A3n \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
Am1 & Am2 & Am3 & \cdots & Amn
\end{array}
$$

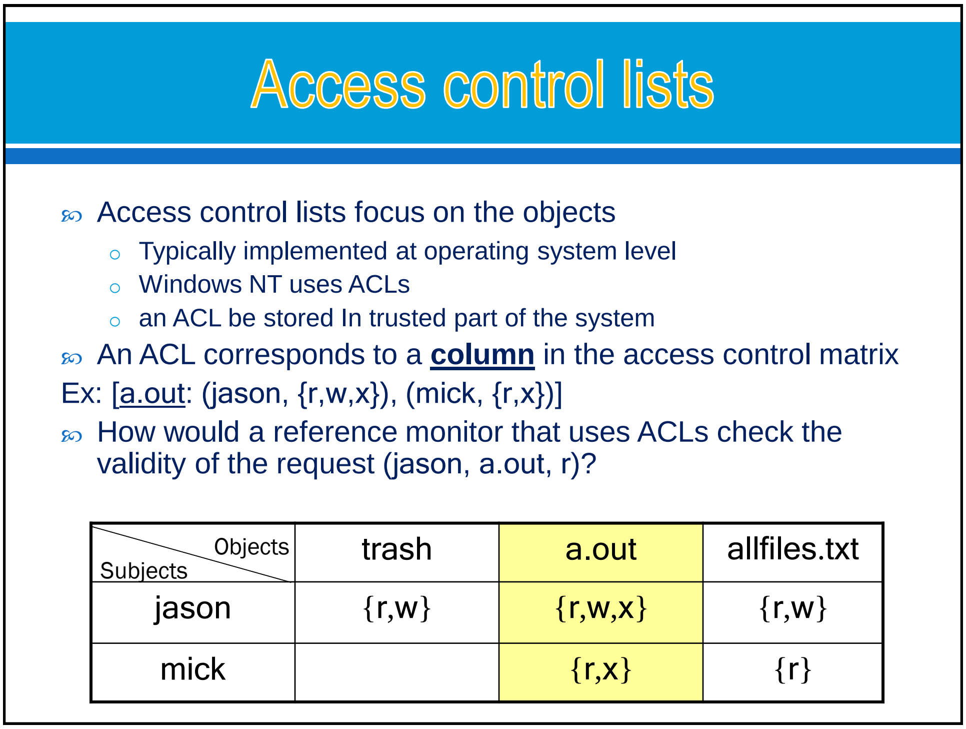
Objects / Subjects	trash	a.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
mick		{r,x}	{r}

- The request (jason, allfiles.txt, w) is granted
- The request (mick, allfiles.txt, w) is denied



Disadvantages

- Abstract formulation of access control
- Not suitable for direct implementation
  - The matrix is likely to be extremely sparse and therefore implementation is inefficient
  - Management of the matrix is likely to be extremely difficult if there are 0000s of files and 00s of users (resulting in 000000s of matrix entries)



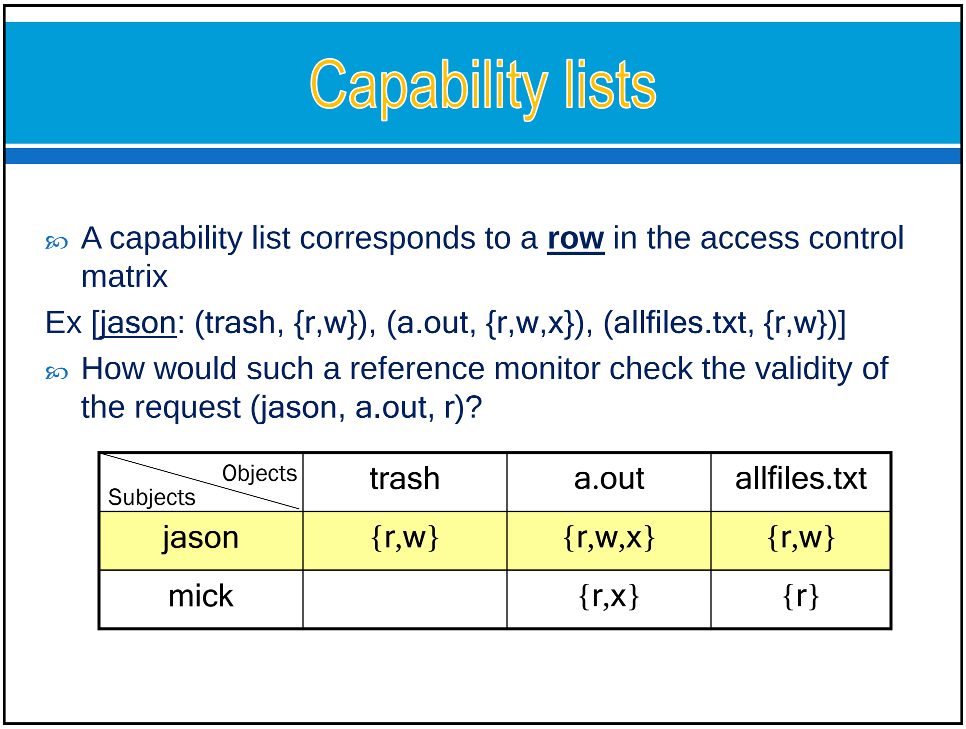
Access control lists

- Access control lists focus on the objects
  - Typically implemented at operating system level
  - Windows NT uses ACLs
  - an ACL be stored In trusted part of the system
- An ACL corresponds to a **column** in the access control matrix

Ex: [a.out: (jason, {r,w,x}), (mick, {r,x})]

- How would a reference monitor that uses ACLs check the validity of the request (jason, a.out, r)?

Objects Subjects	trash	a.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
mick		{r,x}	{r}



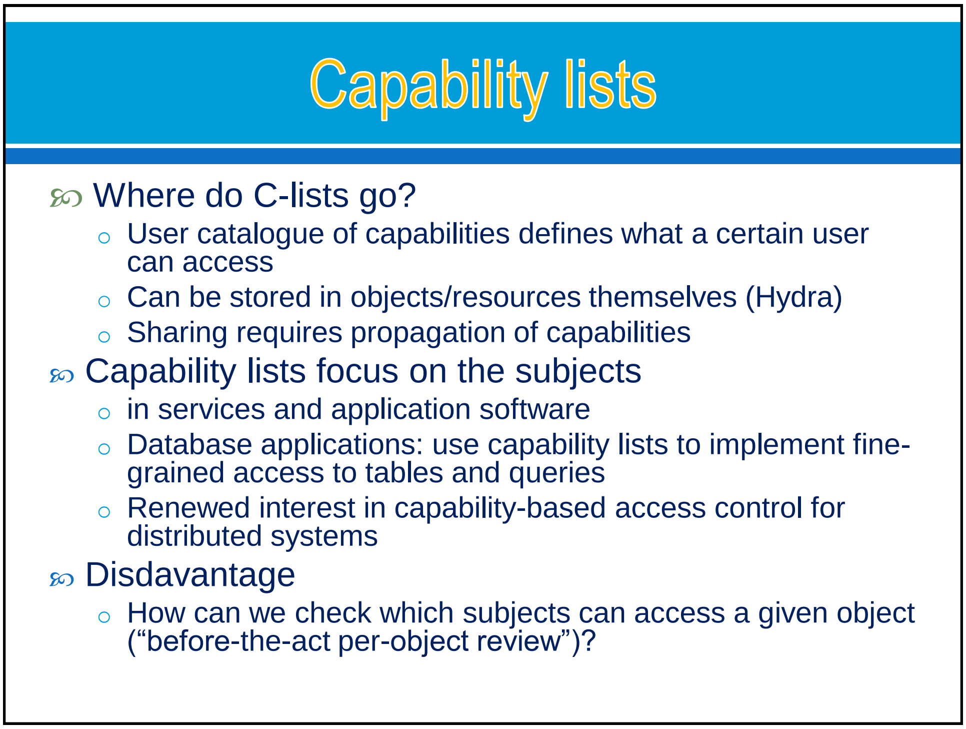
Capability lists

- A capability list corresponds to a **row** in the access control matrix

Ex [jason: (trash, {r,w}), (a.out, {r,w,x}), (allfiles.txt, {r,w})]

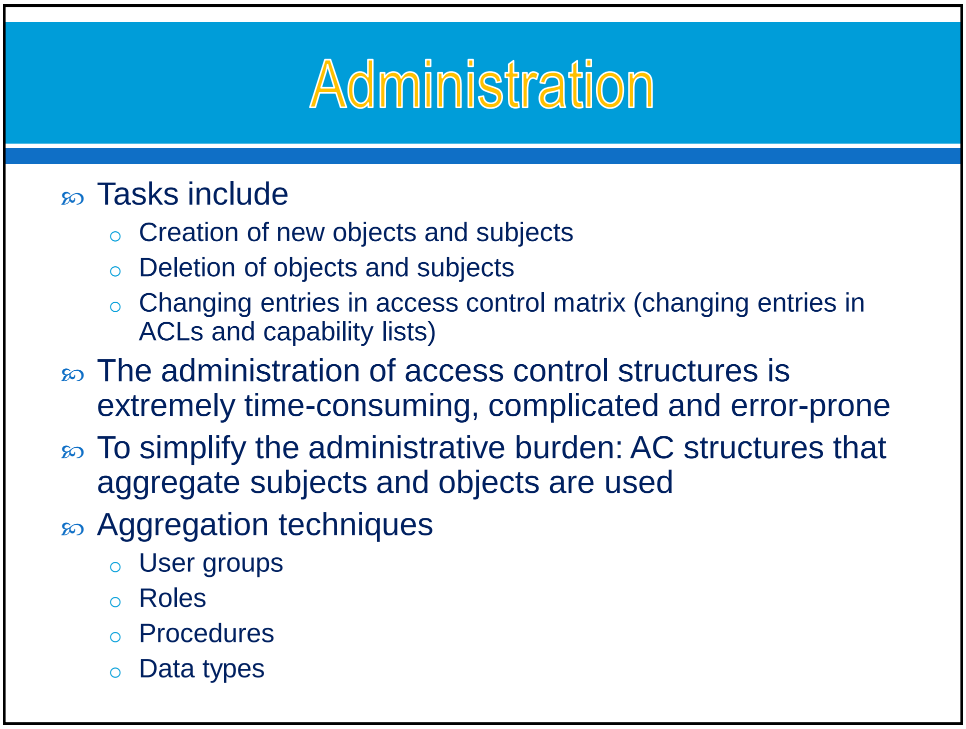
- How would such a reference monitor check the validity of the request (jason, a.out, r)?

Objects Subjects	trash	a.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
mick		{r,x}	{r}



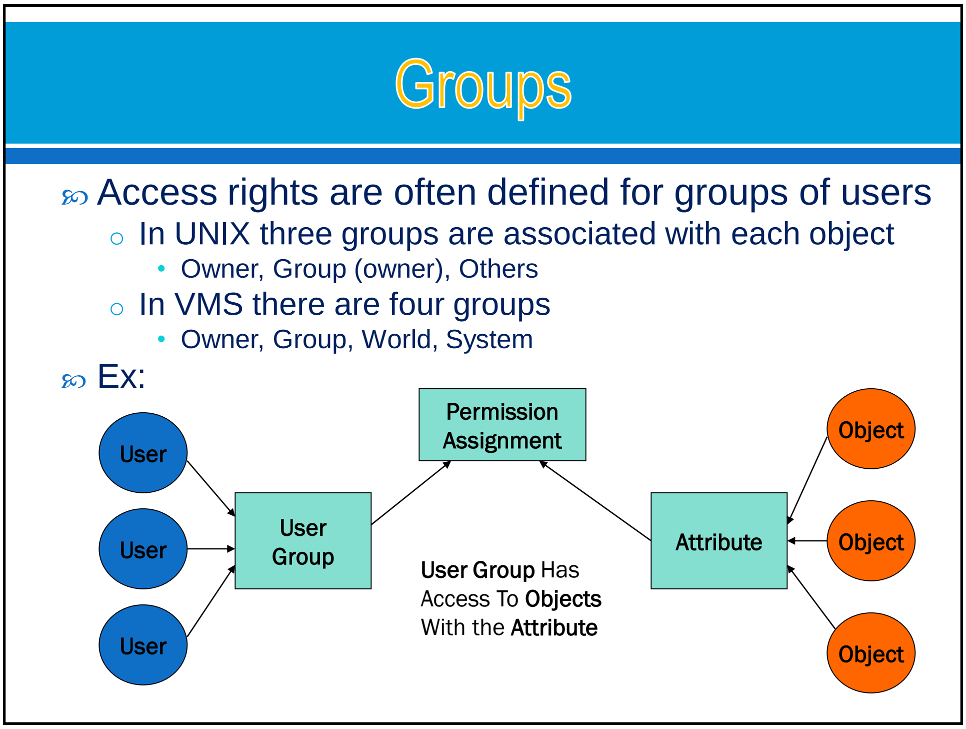
Capability lists

- Where do C-lists go?
  - User catalogue of capabilities defines what a certain user can access
  - Can be stored in objects/resources themselves (Hydra)
  - Sharing requires propagation of capabilities
- Capability lists focus on the subjects
  - in services and application software
  - Database applications: use capability lists to implement fine-grained access to tables and queries
  - Renewed interest in capability-based access control for distributed systems
- Disdavantage
  - How can we check which subjects can access a given object ("before-the-act per-object review")?



Administration

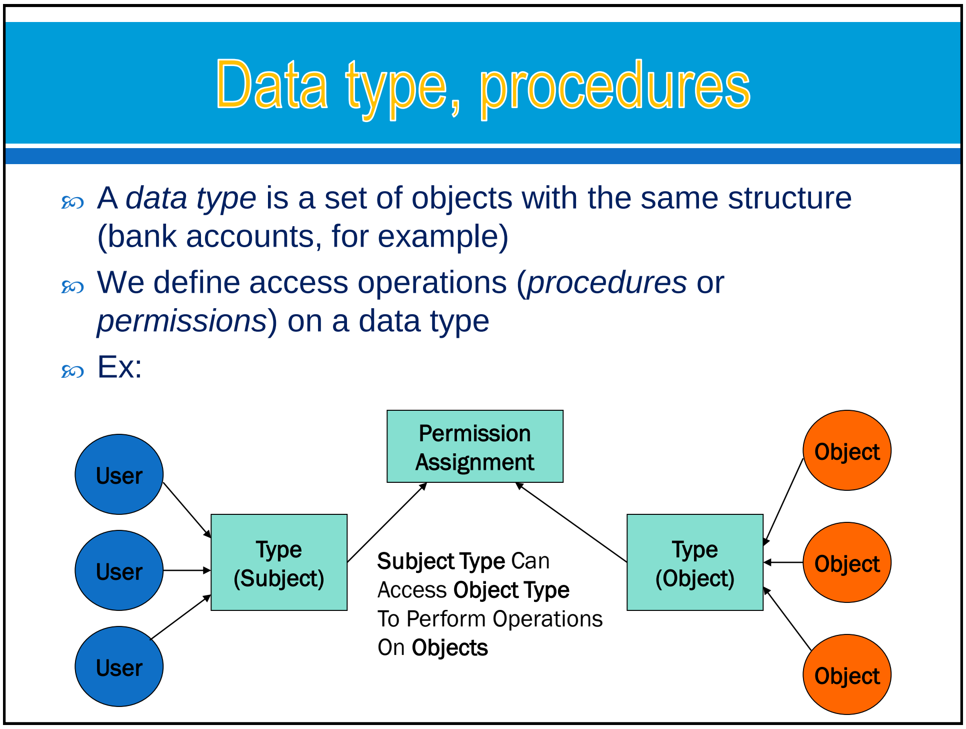
- Tasks include
  - Creation of new objects and subjects
  - Deletion of objects and subjects
  - Changing entries in access control matrix (changing entries in ACLs and capability lists)
- The administration of access control structures is extremely time-consuming, complicated and error-prone
- To simplify the administrative burden: AC structures that aggregate subjects and objects are used
- Aggregation techniques
  - User groups
  - Roles
  - Procedures
  - Data types



Groups

- Access rights are often defined for groups of users
  - In UNIX three groups are associated with each object
    - Owner, Group (owner), Others
  - In VMS there are four groups
    - Owner, Group, World, System
- Ex:

User Group Has Access To **Objects** With the **Attribute**



Data type, procedures

- A *data type* is a set of objects with the same structure (bank accounts, for example)
- We define access operations (*procedures* or *permissions*) on a data type
- Ex:

User

User

User

Type (Subject)

Permission Assignment

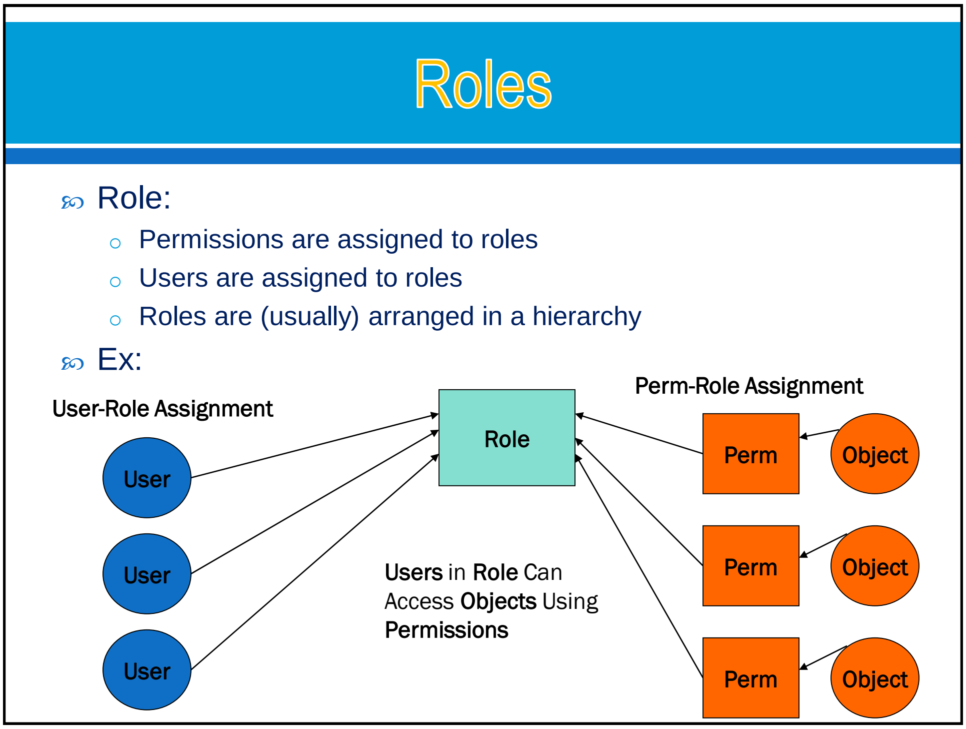
Subject Type Can Access **Object Type** To Perform Operations On **Objects**

Type (Object)

Object

Object

Object



Roles

- Role:
  - Permissions are assigned to roles
  - Users are assigned to roles
  - Roles are (usually) arranged in a hierarchy
- Ex:

User-Role Assignment

User

User

User

Role

Users in **Role** Can Access **Objects** Using **Permissions**

Perm-Role Assignment

Perm

Perm

Perm

Object

Object

Object

Access Control Policies

Access Control

- Many OS to determine whether users are authorized to conduct different actions
  - **the mandatory access control (MAC):** computer system

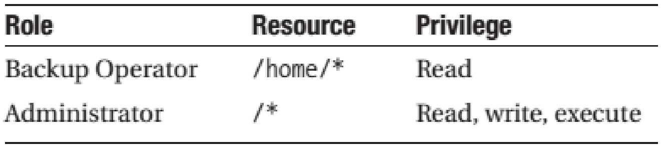
    the computer system decides exactly who has access to which resource in the system
  - **the discretionary access control (DAC)**: users

    users are authorized to determine which other users can access files or other resources that they create **như GG Drive**
  - **the role-based access control (RBAC)**: MAC in special

    the system decides exactly which users are allowed to access which resources—but the system does this in a special way
  - **Attribute-based access control (ABAC):** Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions.
  - **Physical Access Control**: Locks, fences, biometrics, badges, keys
- The Bell-LaPadula Model: certain level of access.

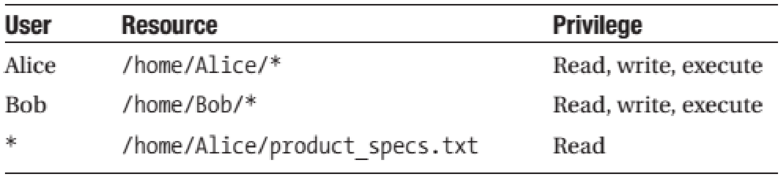
ACL, ex

MAC



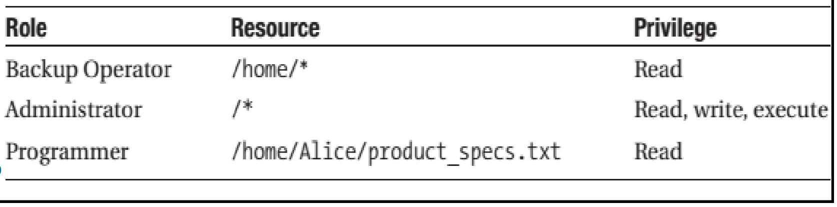
Role	Resource	Privilege
Backup Operator	/home/*	Read
Administrator	/*	Read, write, execute

DAC



User	Resource	Privilege
Alice	/home/Alice/*	Read, write, execute
Bob	/home/Bob/*	Read, write, execute
*	/home/Alice/product_specs.txt	Read

RBAC



Role	Resource	Privilege
Backup Operator	/home/*	Read
Administrator	/*	Read, write, execute
Programmer	/home/Alice/product_specs.txt	Read

Access Control Techniques

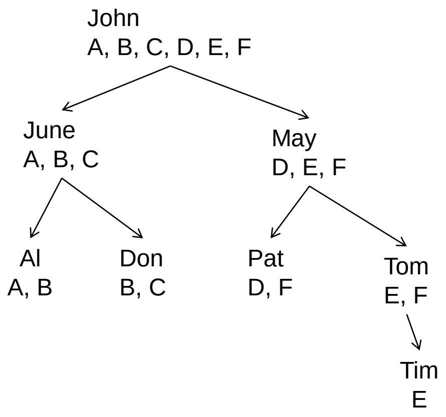
Mandatory Access Control

Login	User	Group	Permi…
John	John	Mgmt	rwx r x
June	June	Billing	r
May	May	Factory	r x r x
Al	Al	Billing	
Don	Don	Billing	

Role-Based Access Control

Login	Role	Permission
John	Mgr	A, B,C,D,E,F
June	Acct.	A,B,C
Al	Acct.	A,B,C
May	Factory	D,E,F
Pat	Factory	D,E,F

Discretionary Access Control



DAC - Discretionary Access Control

Tự quyết định mình sẽ chia sẻ tài nguyên nào cho ai

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Discretionary Access Control

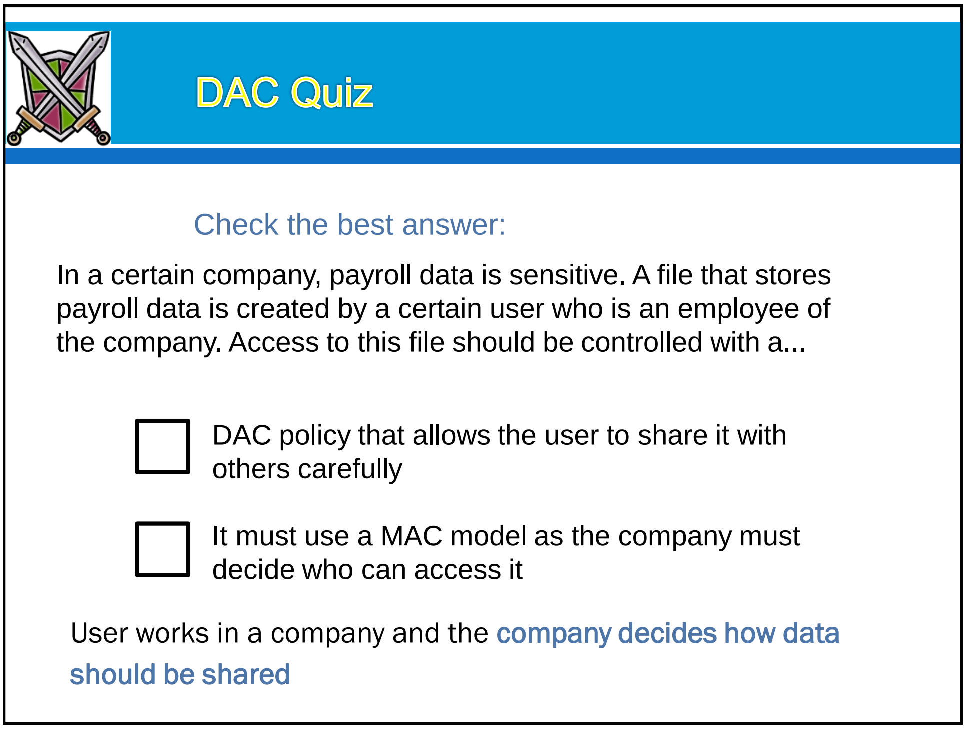
In discretionary access control (DAC), **owner of a resource decides** how it can be shared

- Owner can choose to give **read or write access to other users**

Discretionary Access Control

Two problems with DAC:

- You cannot control if someone you share a file with will not further share the data contained in it
  - **Cannot control "information flow"**
- In many organizations, **a user does not get to decide how certain type of data can be shared**
  - Typically the employer may mandate how to share various types of sensitive data
- Mandatory Access Control (MAC) helps address these problems



DAC Quiz

Check the best answer:

In a certain company, payroll data is sensitive. A file that stores payroll data is created by a certain user who is an employee of the company. Access to this file should be controlled with a...

- □ DAC policy that allows the user to share it with others carefully
- □ It must use a MAC model as the company must decide who can access it

User works in a company and the company decides how data should be shared

MAC - Mandatory Access Control

Quản lý hệ thống thông qua các chương trình đã setup trên máy tính

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Mandatory Access Control (MAC) Models

Military and intelligence agencies:

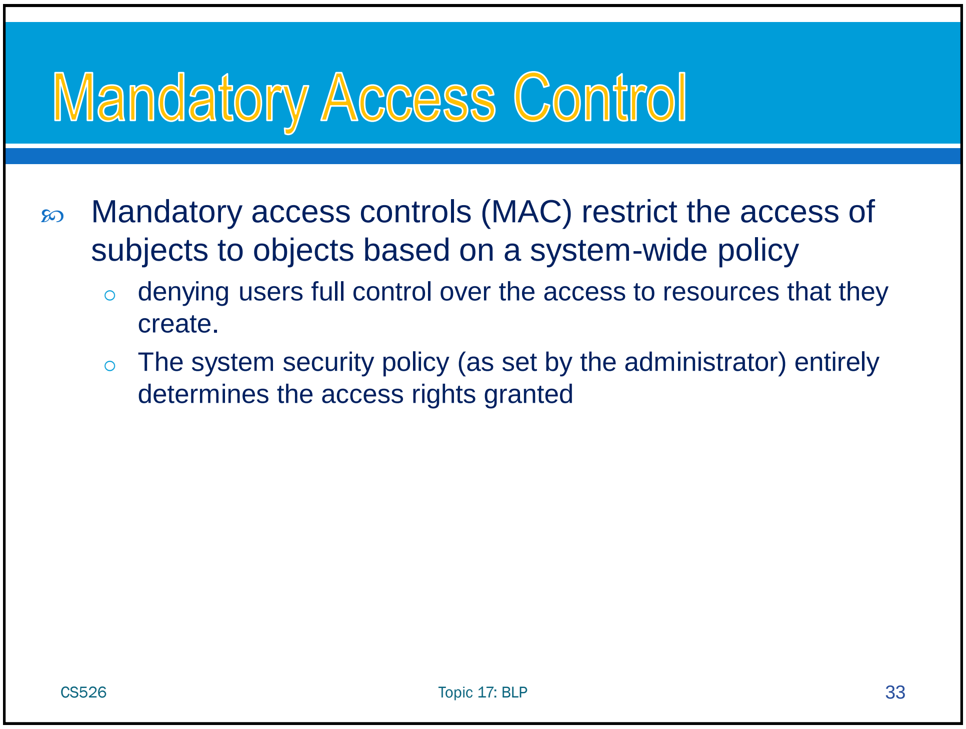
Data has **associated classification level** and users are cleared at various levels

- Top secret, secret, confidential etc.
- Limits on **who can access data at a certain level**
  - User cleared only at secret level should not be able to access top secret data
- Also called **multilevel security (MLS)**

Multi-level Security (MLS)

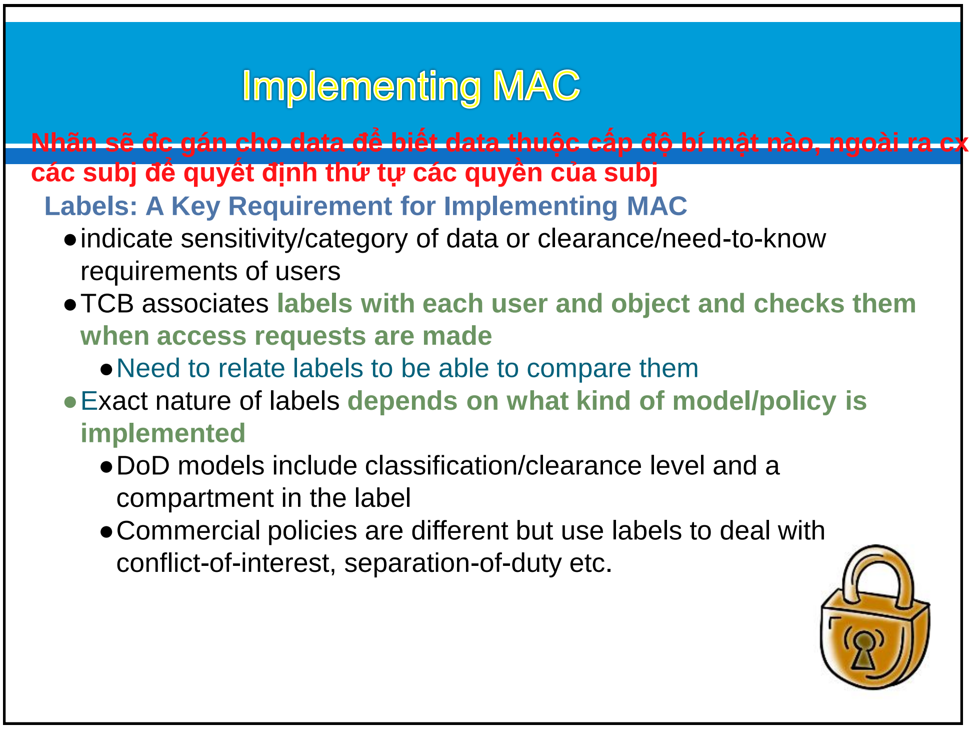
- The capability of a computer system to:
  - carry information with different sensitivities (i.e. classified information at different security levels),
  - permit simultaneous access by users with different security clearances and needs-to-know, and
  - prevent users from obtaining access to information for which they lack authorization.
- Discretionary access control fails to achieve MLS
- Typically use Mandatory Access Control
  - ➔ Primary Security Goal: Confidentiality

32



Mandatory Access Control

- Mandatory access controls (MAC) restrict the access of subjects to objects based on a system-wide policy
  - denying users full control over the access to resources that they create.
  - The system security policy (as set by the administrator) entirely determines the access rights granted



Implementing MAC

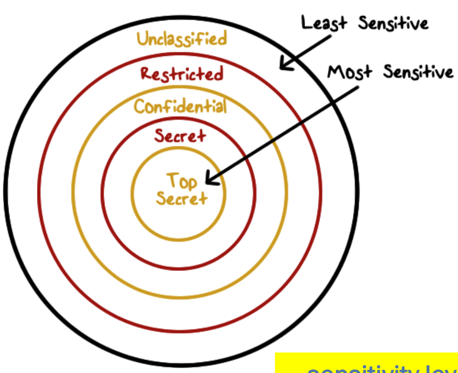
Nhãn sẽ đc gán cho data để biết data thuộc cấp độ bí mật nào, ngoài ra cx đc gán cho các subj để quyết định thứ tự các quyền của subj

Labels: A Key Requirement for Implementing MAC

- indicate sensitivity/category of data or clearance/need-to-know requirements of users
- TCB associates **labels with each user and object and checks them when access requests are made**
  - Need to relate labels to be able to compare them
- Exact nature of labels **depends on what kind of model/policy is implemented**
  - DoD models include classification/clearance level and a compartment in the label
  - Commercial policies are different but use labels to deal with conflict-of-interest, separation-of-duty etc.

Multilevel in Implementing MAC

Example of Labels/MAC in a DoD Environment:



- Label = (sensitivity level, Compartments)
- Let us consider highly sensitive documents that have information about various arms stockpiles.

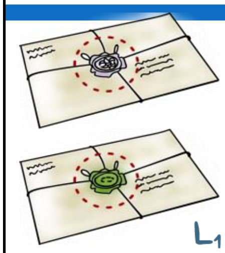
  L1 = (TS, {nuclear, chemical})
  L2 = (S, {nuclear, conventional})

Các nhãn này có thể so sánh được với nhau

- Providing confidential access to documents (Bell and La Padula or BLP Model)

sensitivity levels are totally ordered
(TS > S > C > U)

Comparing Labels



- Compartments are sets which can only be partially ordered
- How do we order labels?

$L_1 = (X_1, Comp_1),\ L_2 = (X_2, Comp_2)$

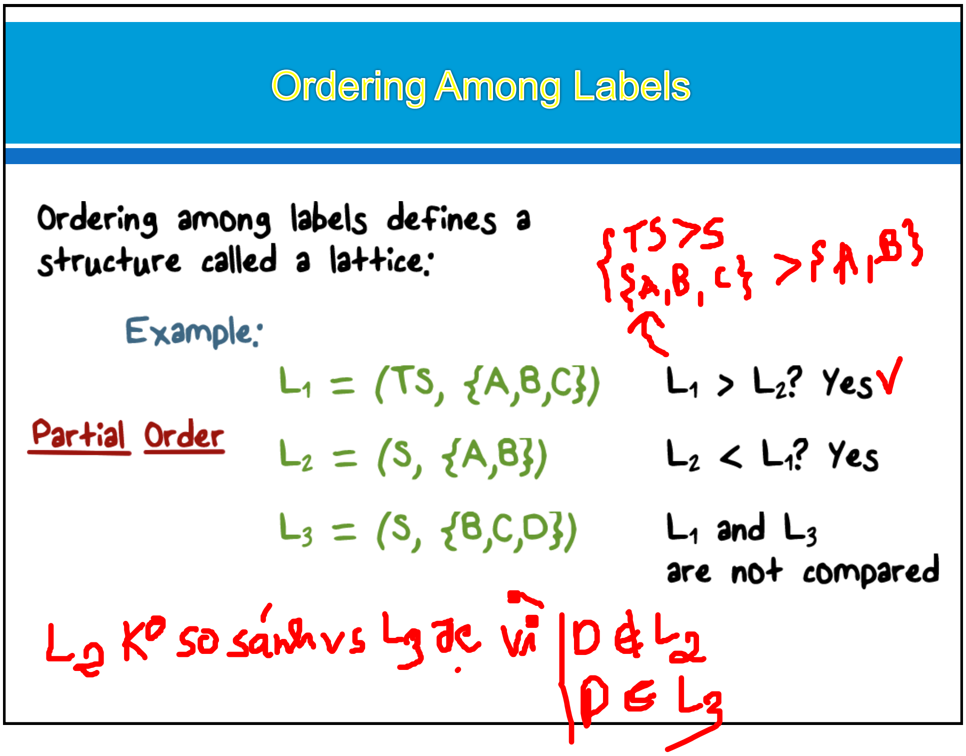
Thống trị

L₁ dominates L₂ : $l_1 > l_2$ and $Comp_1 \geq Comp_2$

or L₁ is dominated by L₂ : $l_1 < l_2$ and $Comp_1 \leq Comp_2$

or $L_1 = L_2$: $l_1 = l_2$ and $Comp_1 = Comp_2$

or L₁ and L₂ are not comparable : $L_1 \ngtr L_2$ and $L_1 \nless L_2$ and $L_1 \neq L_2$



Ordering Among Labels

Ordering among labels defines a structure called a lattice:

{TS > S
{{A, B, C} > {A, B}
↑

Example:

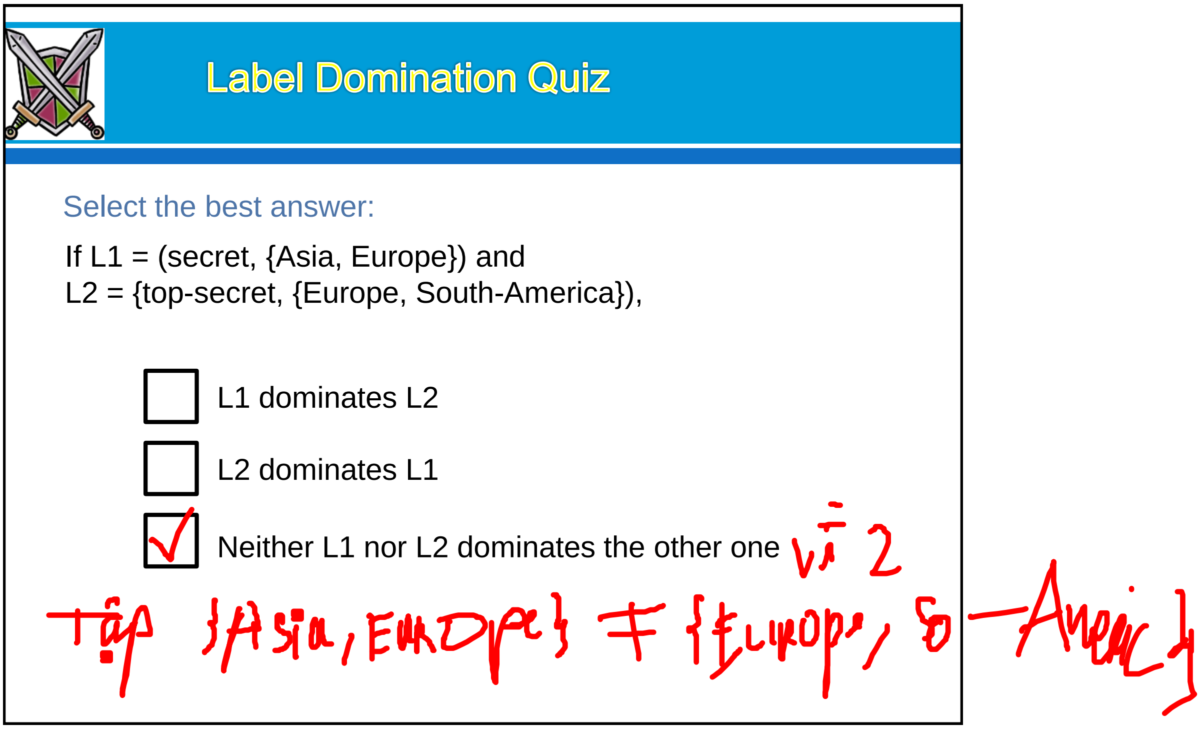
Partial Order

$L_1 = (TS, \{A,B,C\})$ $L_1 > L_2$? Yes ✓

$L_2 = (S, \{A,B\})$ $L_2 < L_1$? Yes

$L_3 = (S, \{B,C,D\})$ L_1 and L_3 are not compared

L_2 Kº so sánh vs L_3 do vì | $D \notin L_2$
| $D \in L_3$



Label Domination Quiz

Select the best answer:

If L1 = (secret, {Asia, Europe}) and
L2 = {top-secret, {Europe, South-America}),

☐ L1 dominates L2

☐ L2 dominates L1

☑ Neither L1 nor L2 dominates the other one vì 2

Tập {Asia, Europe} ≠ {Europe, So-America}

Sensitive Data Quiz

Select the best answer:

Assume that label L1 or a document D1 dominates label L2 of document D2 when these labels are defined by (sensitivity level, compartment).

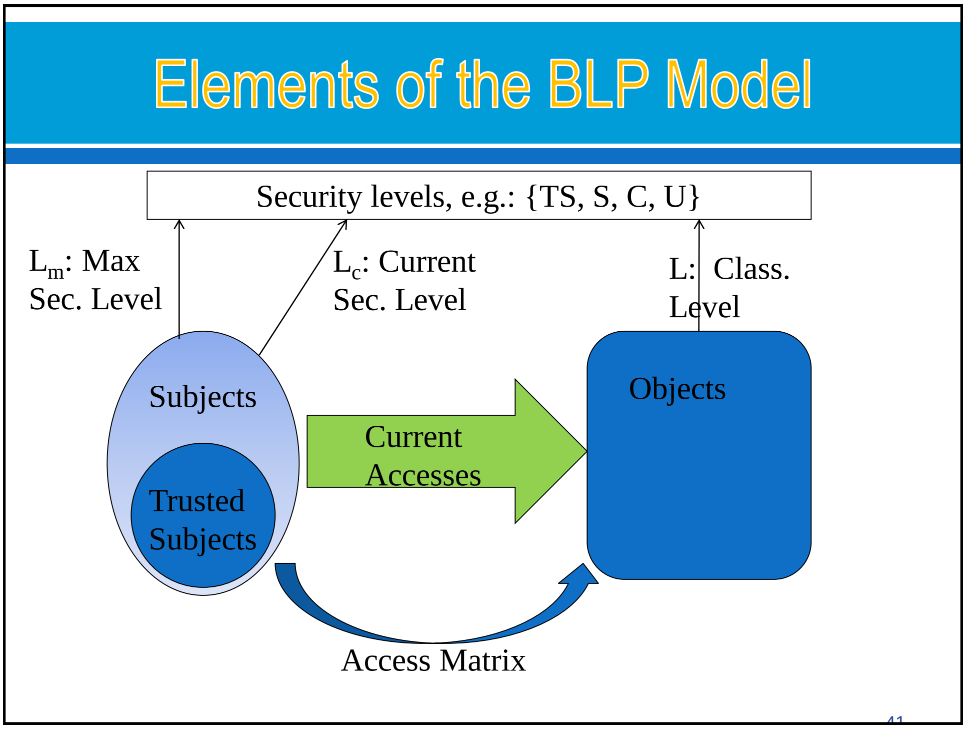
□ D1 contains more sensitive data than D2.

□ D2 is more sensitive than D1.

□ The data contained in D2 has a narrower scope as defined by its compartment

Bell-LaPadula Model: A MAC Model for Achieving Multi-level Security

- Introduce in 1973
- Air Force was concerned with security in time-sharing systems
  - Many OS bugs
  - Accidental misuse
- Main Objective:
  - Enable one to formally show that a computer system can securely process classified information



Elements of the BLP Model

Security levels, e.g.: {TS, S, C, U}

L_m: Max Sec. Level

L_c: Current Sec. Level

L: Class. Level

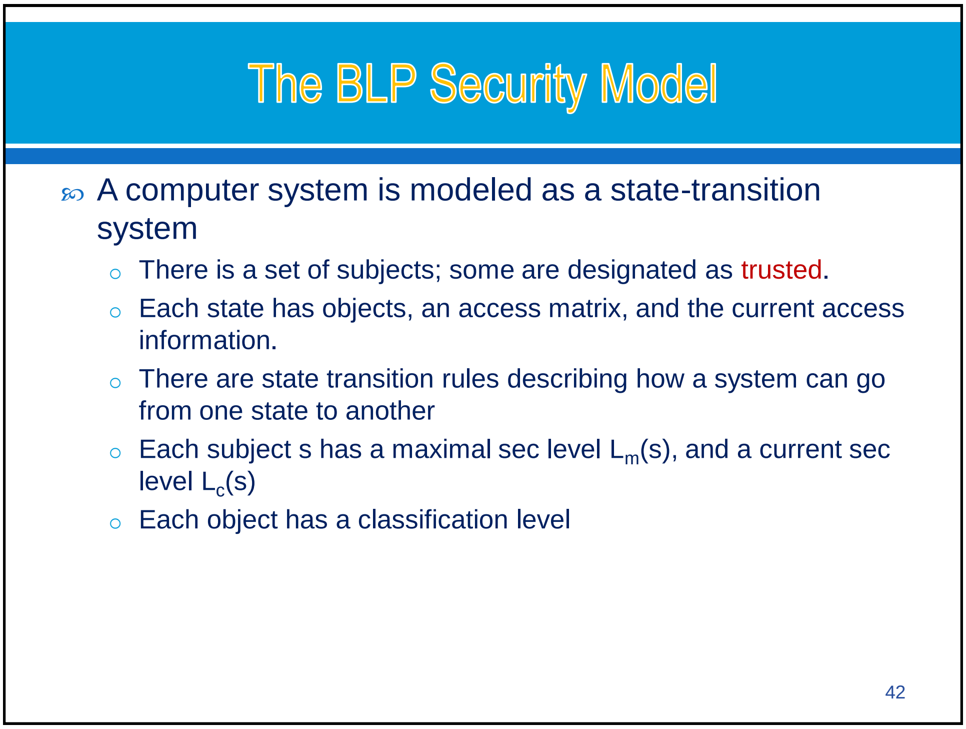
Subjects

Trusted Subjects

Current Accesses

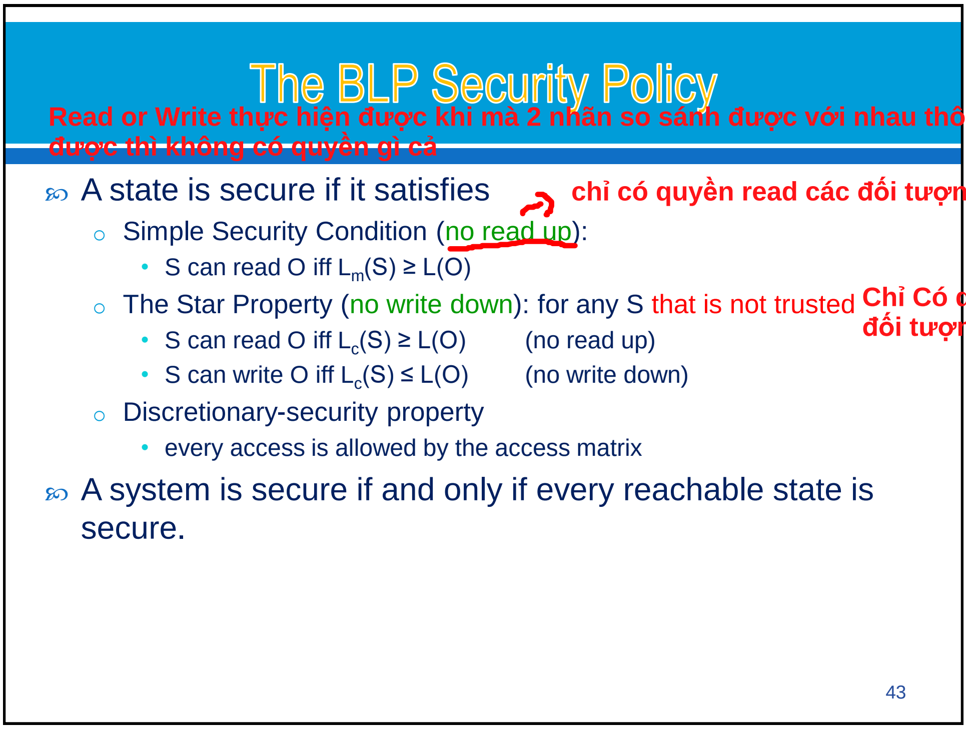
Objects

Access Matrix



The BLP Security Model

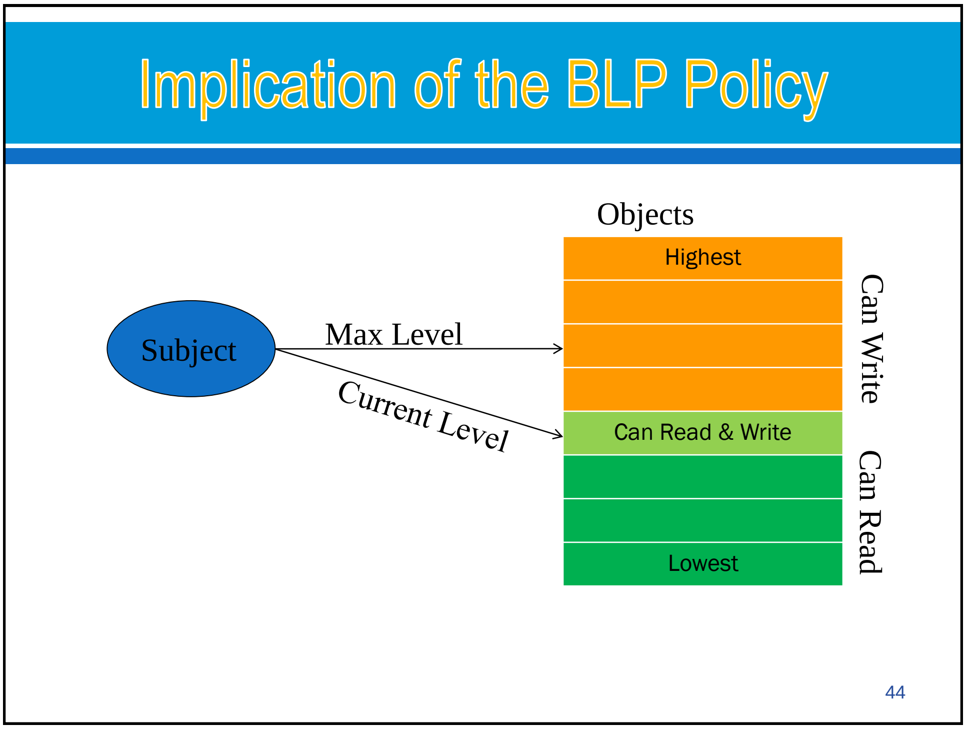
- A computer system is modeled as a state-transition system
  - There is a set of subjects; some are designated as trusted.
  - Each state has objects, an access matrix, and the current access information.
  - There are state transition rules describing how a system can go from one state to another
  - Each subject s has a maximal sec level $L_m(s)$, and a current sec level $L_c(s)$
  - Each object has a classification level



The BLP Security Policy

Read or Write thực hiện được khi mà 2 nhãn so sánh được với nhau thôi, ko so sánh được thì không có quyền gì cả

- A state is secure if it satisfies
  - Simple Security Condition (no read up): chỉ có quyền read các đối tượng quyền thấp hơn
    - S can read O iff $L_m(S) \geq L(O)$
  - The Star Property (no write down): for any S that is not trusted Chỉ Có quyền write lên đối tượng cao hơn
    - S can read O iff $L_c(S) \geq L(O)$ (no read up)
    - S can write O iff $L_c(S) \leq L(O)$ (no write down)
  - Discretionary-security property
    - every access is allowed by the access matrix
- A system is secure if and only if every reachable state is secure.



Implication of the BLP Policy

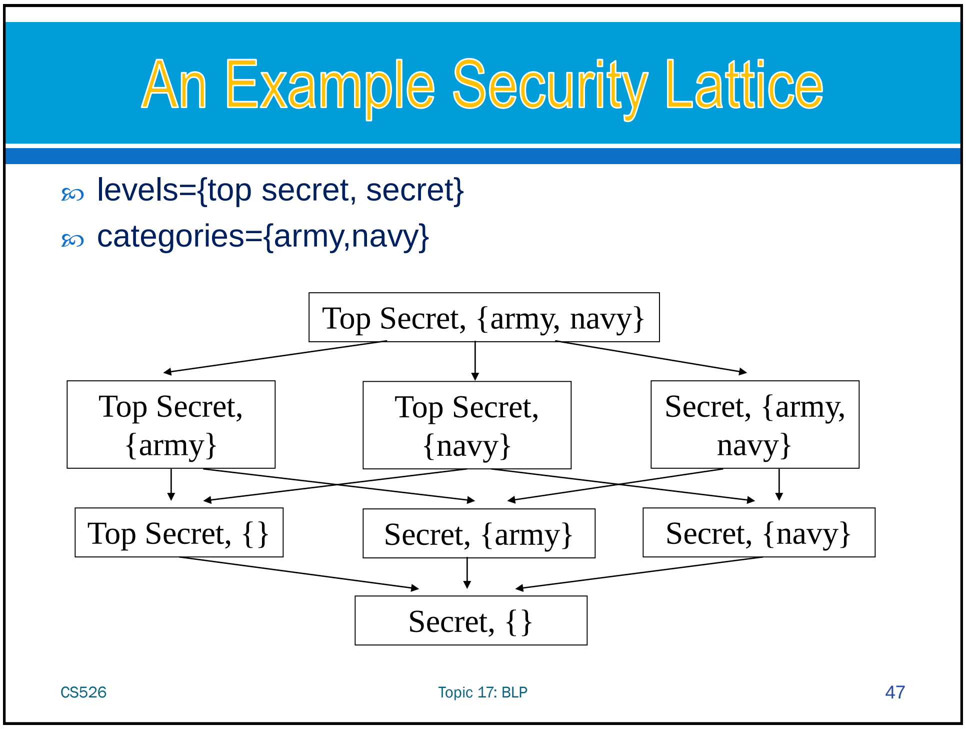
Star-property

- Applies to subjects (principals) not to users
- Users are trusted (must be trusted) not to disclose secret information outside of the computer system
- Subjects are not trusted because they may have Trojan Horses embedded in the code they execute
- Star-property prevents overt leakage of information and does not address the covert channel problem

More on MLS: example

- Used as attributes of both subjects & objects
  - clearance & classification
- MLS
  - security levels
  - security categories: Also known as compartments
  - security labels = Levels × P (Categories)
- Define an ordering relationship among Labels
  - (e1, C1) ≤ (e2, C2) iff. e1 ≤e2 and C1 ⊆ C2

Apply	security levels	security categories
military	top secret ≥ secret ≥ confidential ≥ unclassified	• army, navy, air force • nato, nasa, noforn
commercial	restricted ≥ proprietary ≥ sensitive ≥ public	• Sales, R&D, HR • Dept A, Dept B, Dept C



An Example Security Lattice

- levels={top secret, secret}
- categories={army,navy}

Top Secret, {army, navy}

Top Secret, {army}

Top Secret, {navy}

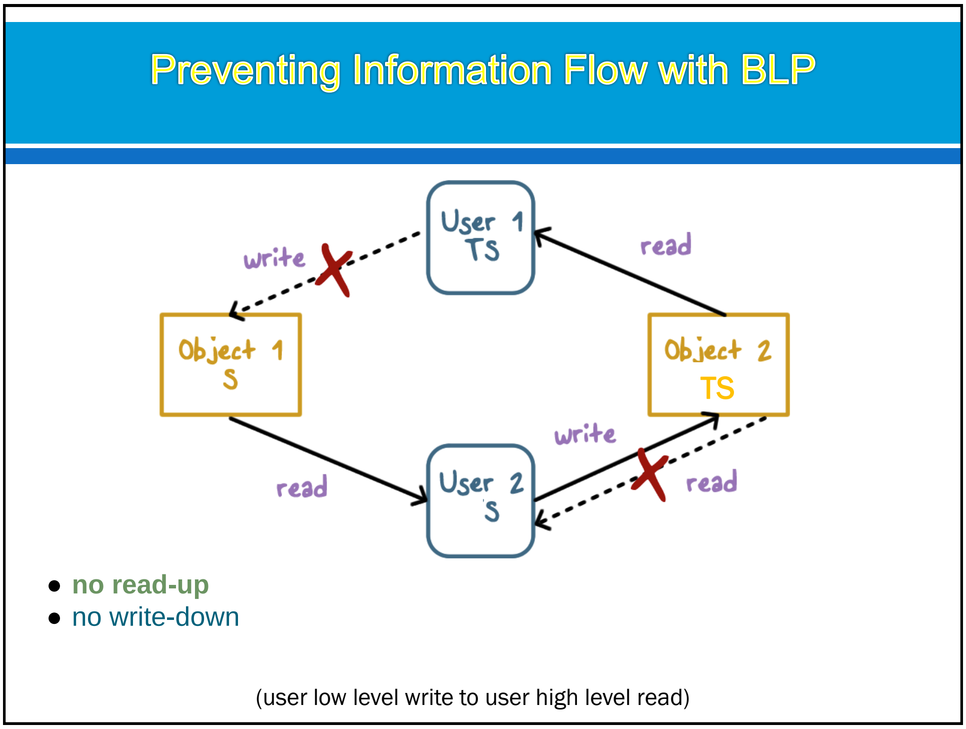
Secret, {army, navy}

Top Secret, {}

Secret, {army}

Secret, {navy}

Secret, {}



Preventing Information Flow with BLP

- no read-up
- no write-down

(user low level write to user high level read)

Classified Documents Quiz

Select the best answer:

Since an unclassified document contains no sensitive information, it can be read or written by anyone in a system that implements BLP

□ True

□ False

BLP allows an unclassified user to write a top secret document.

□ True

□ False

1. {TopSecret > Secret
{A,B} != {B,C} → Lan(TopSe, {A,B}) <> obj(Sec, {B,C}) → No W, No Read

Example

- Given the security levels TOP SECRET, SECRET, CONFIDENTIAL, and UNCLASSIFIED (ordered from highest to lowest), and the categories A, B, and C, specify what type of access (read, write, or both) is allowed in each of the following situations. Assume that discretionary access controls allow anyone access unless otherwise specified.
  - Paul, cleared for (TOP SECRET, {A, C}), wants to access a document classified (SECRET, { B, C }). **No Read, No Write**
  - Anna, cleared for (CONFIDENTIAL, {C}), wants to access a document classified (CONFIDENTIAL, {B}). **Read, No Write**
  - Jesse, cleared for (SECRET, {C}), wants to access a document classified (CONFIDENTIAL, {C}). **No Read, No Write**
  - Sammi, cleared for (TOP SECRET, {A, C}), wants to access a document classified (CONFIDENTIAL, {A}). **Read & Write**
  - Robin, who has no clearances (and so works at the UNCLASSIFIED level), wants to access a document classified (CONFIDENTIAL, {B}). **No read, Write**

5. {Unclass < Confidential
{∅} ≠ {B} → W, no Read

Other MAC Models

- **Biba is dual of BLP**
  - Focuses on **integrity rather than confidentiality**
  - Read-up and write-down rules

Example:

- Integrity level could be **high, medium or low**
- Compartment could be similar to BLP and captures topic(s) of document
- Low integrity information should never flow up into high integrity documents

Policies for Commercial Environments

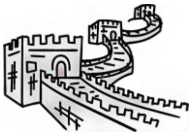
- **User clearance is not common**
- Other requirements exist
  - **Data only be accessed by certain application** (e.g., payroll)
  - **Separation-of-duty and conflict-of-interest requirements**

Policies for Commercial Environments

- Clark-Wilson Policy

  Users ➔ Programs (transactions) ➔ Objects

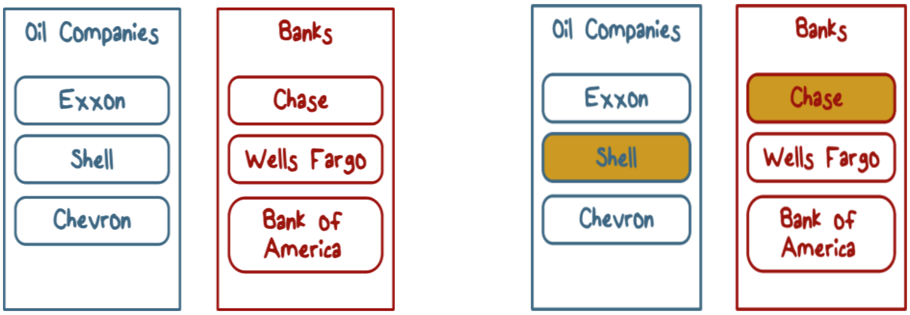
  - Same user cannot execute two programs that require separation-of-duty



- Chinese Wall Policy
  - Deals with conflict of interest

Chinese Wall Policy (Conflict of Interest)

Objects are put into conflict classes:



Oil Companies

Exxon

Shell

Chevron

Banks

Chase

Wells Fargo

Bank of America

Oil Companies

Exxon

Shell

Chevron

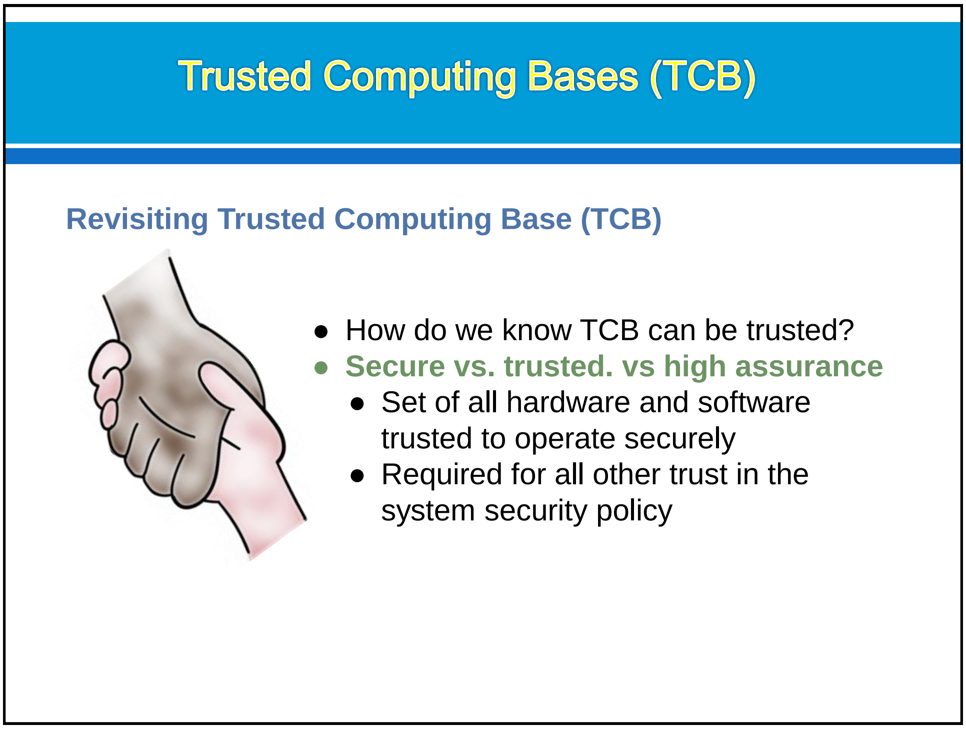
Banks

Chase

Wells Fargo

Bank of America

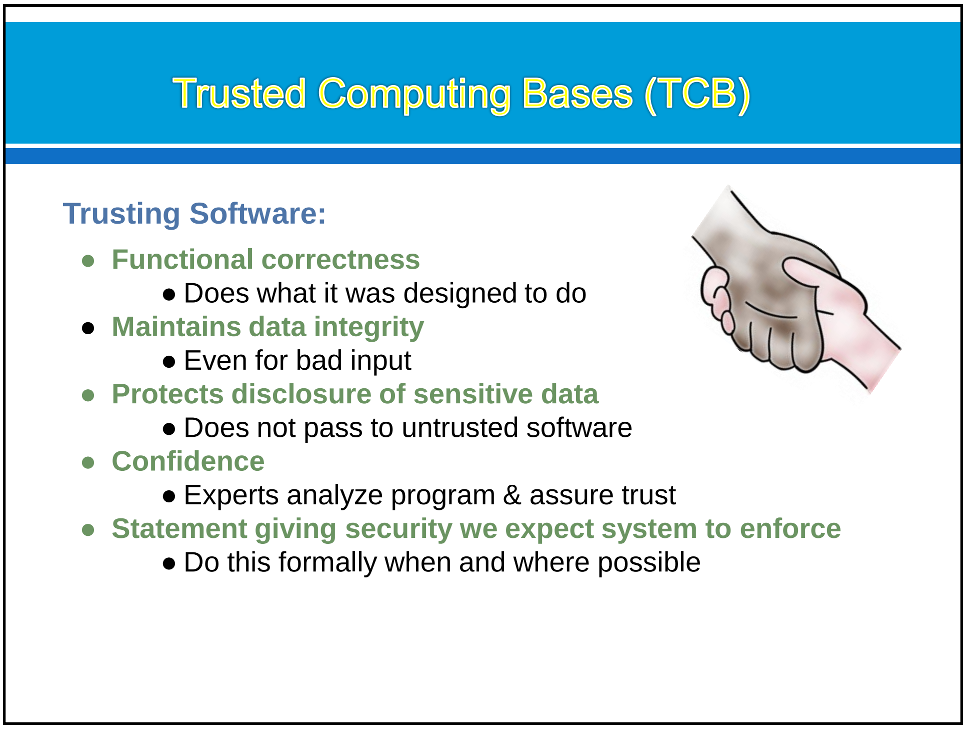
The user **can access any object** as long as he/she has not accessed an object from another company **in the same conflict class**.



Trusted Computing Bases (TCB)

Revisiting Trusted Computing Base (TCB)

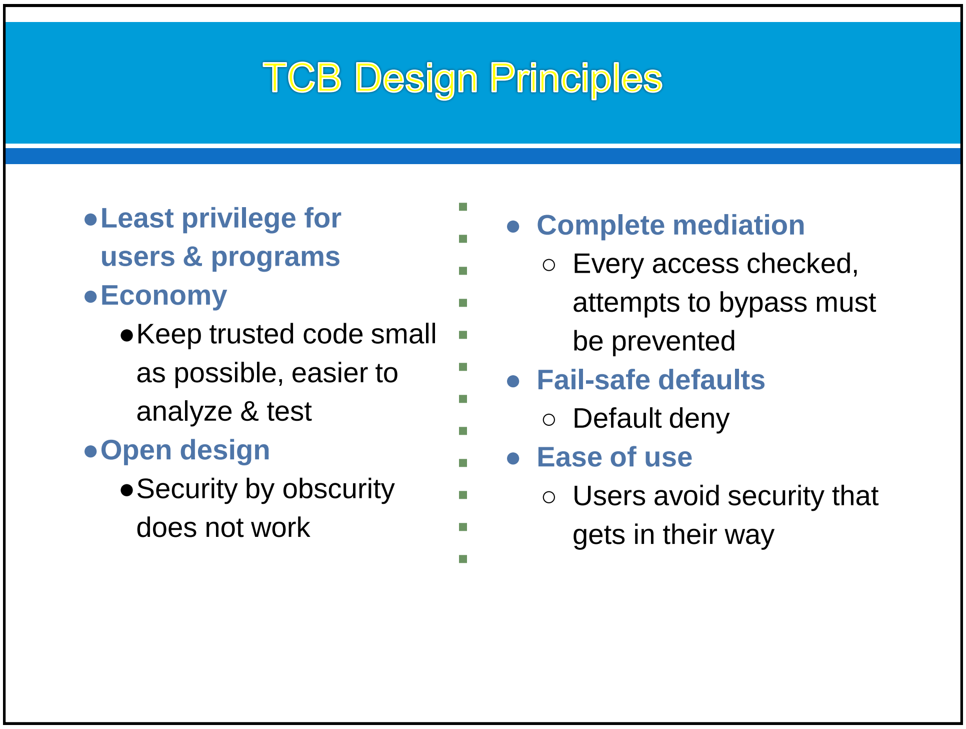
- How do we know TCB can be trusted?
- **Secure vs. trusted. vs high assurance**
  - Set of all hardware and software trusted to operate securely
  - Required for all other trust in the system security policy



Trusted Computing Bases (TCB)

Trusting Software:

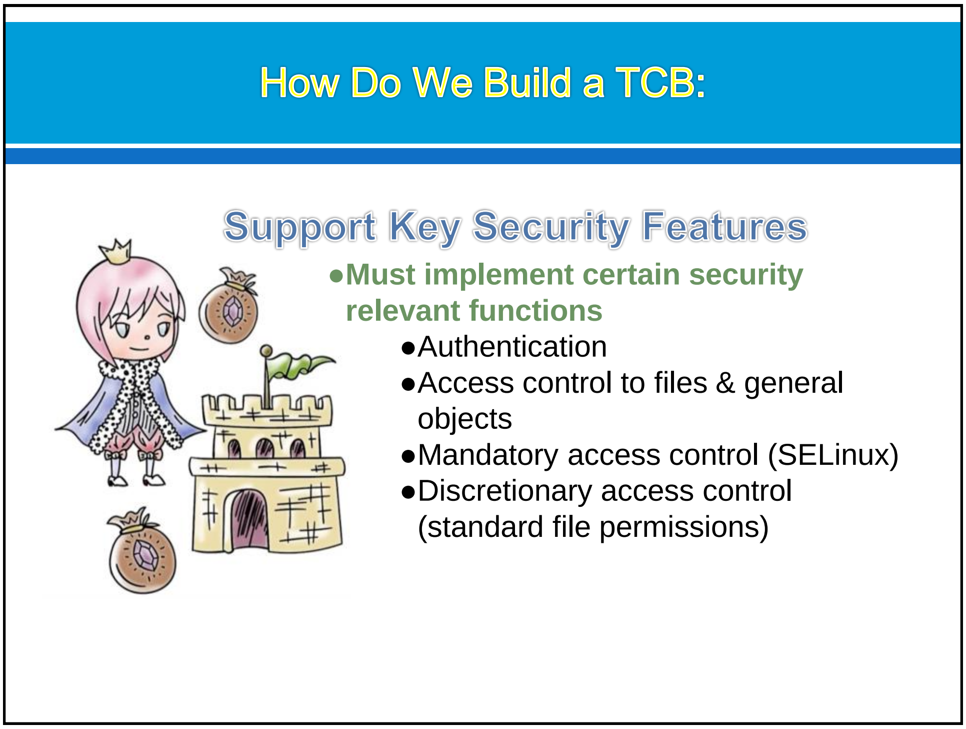
- **Functional correctness**
  - Does what it was designed to do
- **Maintains data integrity**
  - Even for bad input
- **Protects disclosure of sensitive data**
  - Does not pass to untrusted software
- **Confidence**
  - Experts analyze program & assure trust
- **Statement giving security we expect system to enforce**
  - Do this formally when and where possible



TCB Design Principles

- **Least privilege for users & programs**
- **Economy**
  - Keep trusted code small as possible, easier to analyze & test
- **Open design**
  - Security by obscurity does not work

- **Complete mediation**
  - ○ Every access checked, attempts to bypass must be prevented
- **Fail-safe defaults**
  - ○ Default deny
- **Ease of use**
  - ○ Users avoid security that gets in their way



How Do We Build a TCB:

Support Key Security Features

- **Must implement certain security relevant functions**
  - Authentication
  - Access control to files & general objects
  - Mandatory access control (SELinux)
  - Discretionary access control (standard file permissions)

How Do We Build a TCB:

- **Protection of data used by OS** (OS must protect itself)
  - Security features of trusted OSes
    - Object reuse protection
    - Disk blocks, memory frames reused
    - Process can allocate disk or memory, then look to see what's left behind
    - Trusted OS should zero out objects before reuse
    - **Secure file deletion**: overwrite with varying patterns of zeros & ones
    - **Secure disk destruction**: degaussing, physical destruction

How Do We Build a TCB:

- **Complete mediation of accesses**
- Trusted path from user to secure system
  - Prevents programs from spoofing interface of secure components
  - Prevents programs from tapping path (e.g. keyloggers)
- **Audit log showing object accesses** – only useful if you /look/ at the log
  - Detect unusual use of the system

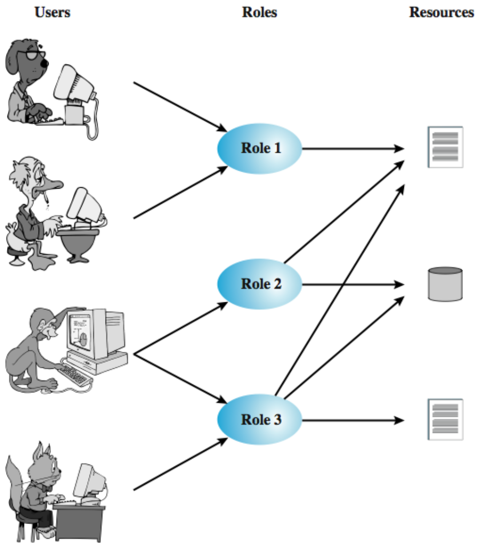
RBAC - role-based access control

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

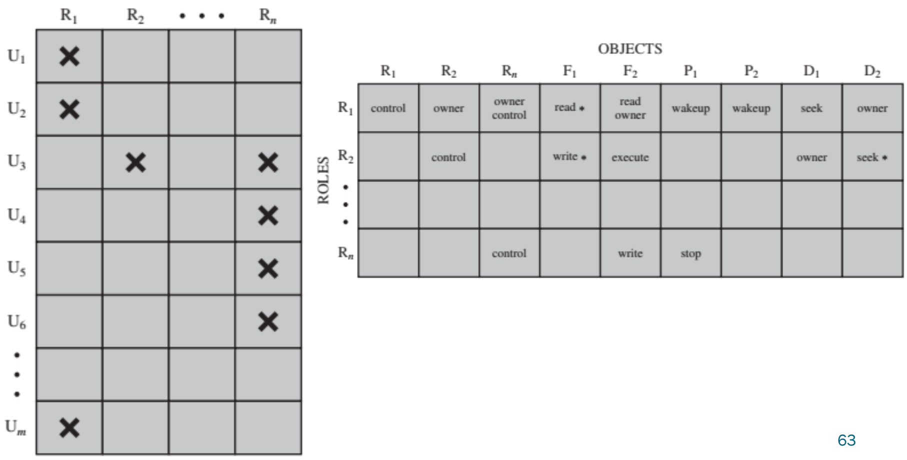
RBAC Model – users, Roles, resources

- RBAC Model:

the system decides exactly which users are allowed to access which resources—but the system does this in a special way



Access Control Matrix Representation of RBAC



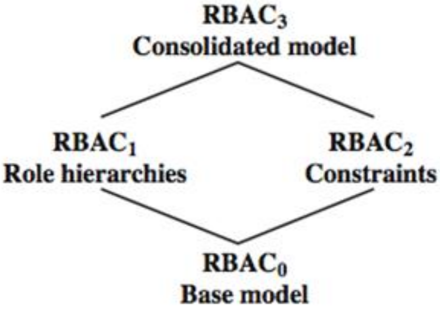
	R_1	R_2	• • •	R_n
U_1	✕			
U_2	✕			
U_3		✕		✕
U_4				✕
U_5				✕
U_6				✕
• • •				
U_m	✕			

OBJECTS

ROLES	R_1	R_2	R_n	F_1	F_2	P_1	P_2	D_1	D_2
R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R_2		control		write *	execute			owner	seek *
• • •									
R_n			control		write	stop			

RBAC Model - Relationship among RBAC models

- $RBAC_0$: the minimum functionality
- $RBAC_1$: the $RBAC_0$ functionality + role hierarchies, which enable one role to inherit permissions from another role.
- $RBAC_2$: $RBAC_0$ + constraints, which restricts the ways in which the components of a RBAC system may be configured.
- $RBAC_3$: $RBAC_0$ + $RBAC_1$ + $RBAC_2$



(a) Relationship among RBAC models

- Constraints provide a means of adapting RBAC to the specifics of administrative and security policies in an organization. A constraint is a defined relationship among roles or a condition related to roles

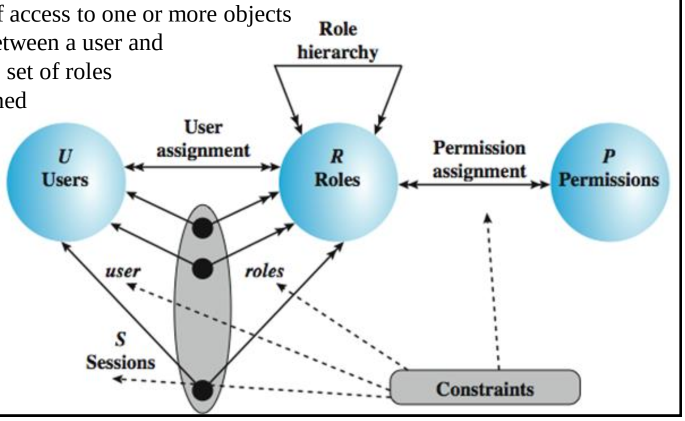
Hierarchies and Constraints

- Role hierarchy
  - Problem: does organizational hierarchy correspond to a permission inheritance hierarchy?
  - Problem: do organizational roles make sense for building hierarchies?
- Constraints
  - Problem: constraints apply to all states, so they require a predicate calculus in general
  - Problem: Only certain types of constraints can effectively be administered? Mutual exclusion, separation of duty, cardinality, etc.
- Conflicts
  - May find other concepts useful for resolving conflicts between constraints and hierarchies/assignments

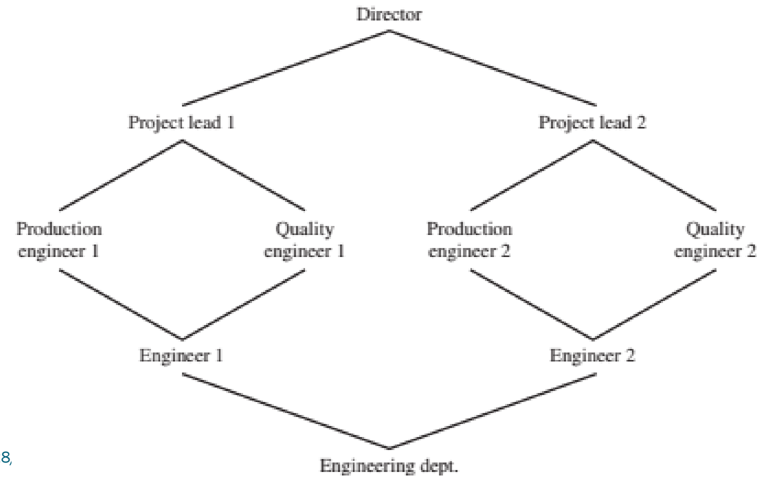
RBAC Model – an RBAC

- An RBAC system contains the four types of entities (the minimum functionality for an RBAC system):

• **User:** An individuals - access to this computer system
• **Role:** job function - controls this computer system
• **Permission:** approval of access to one or more objects
• **Session: :** A mapping between a user and an activated subset of the set of roles to which the user is assigned



Example of Role Hierarchy



Director

Project lead 1

Project lead 2

Production engineer 1

Quality engineer 1

Production engineer 2

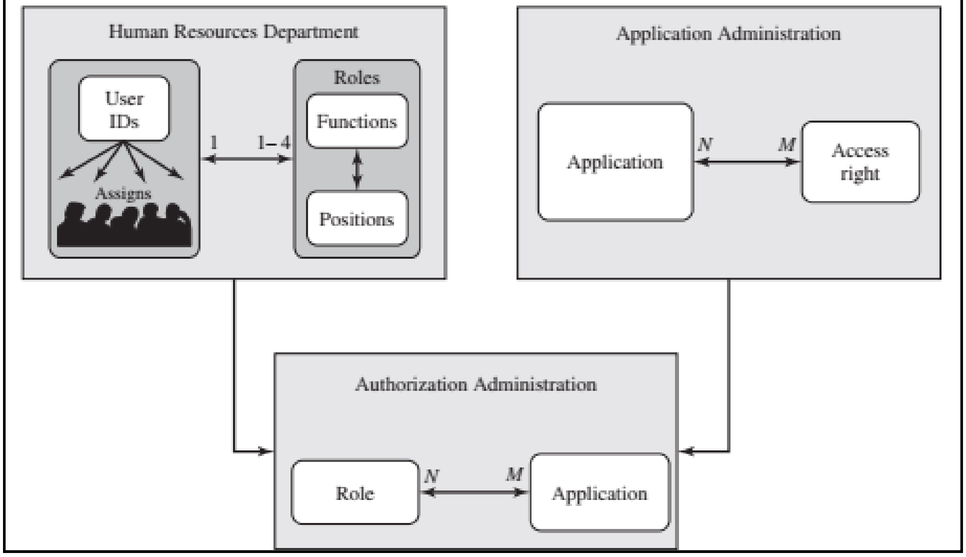
Quality engineer 2

Engineer 1

Engineer 2

Engineering dept.

Example of Access Control Administration



Human Resources Department

User IDs

Assigns

1

1–4

Roles

Functions

Positions

Application Administration

Application

N

M

Access right

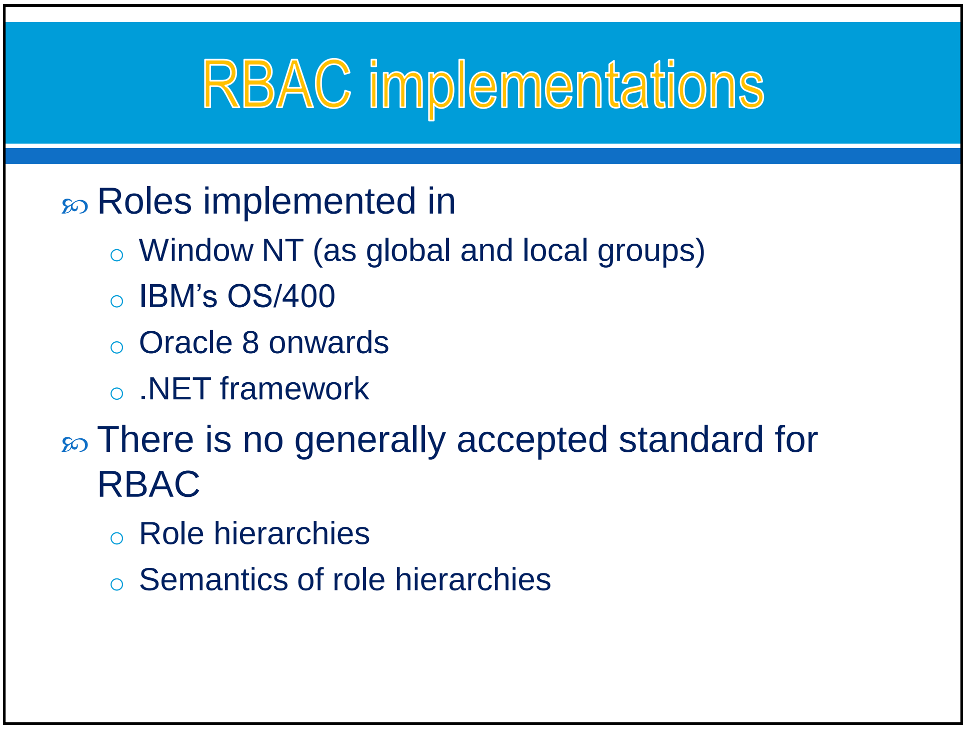
Authorization Administration

Role

N

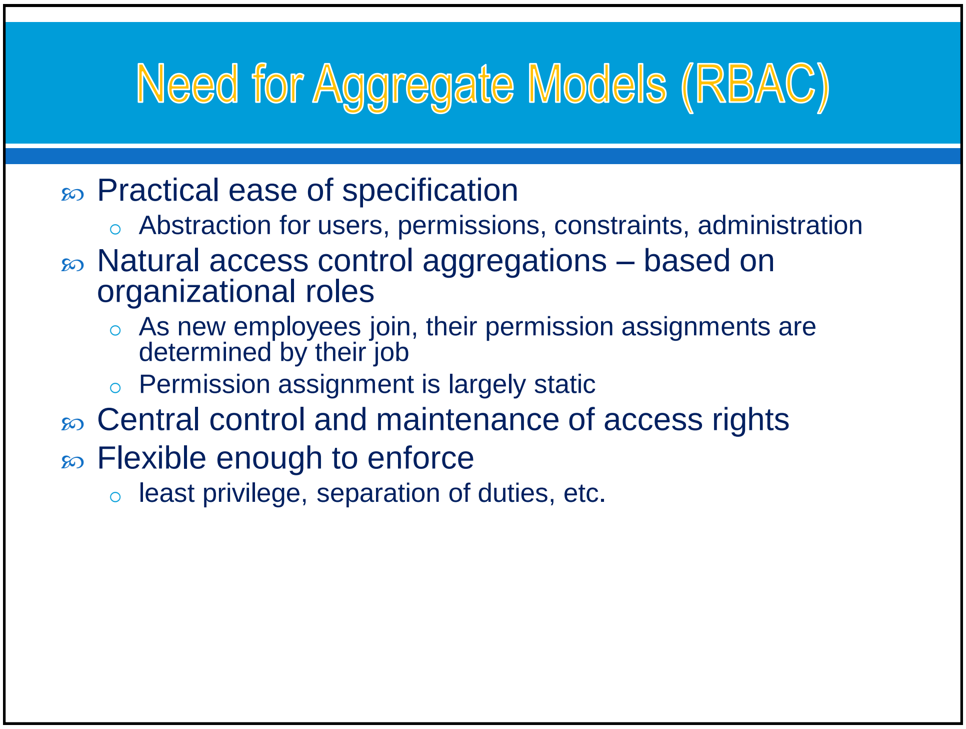
M

Application



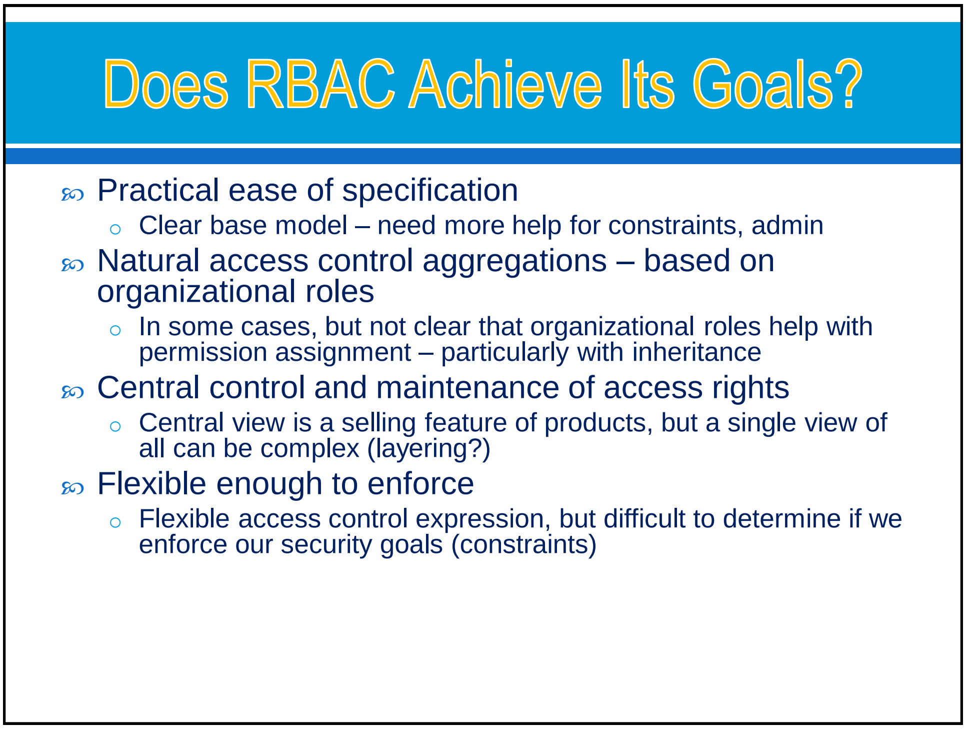
RBAC implementations

- Roles implemented in
  - Window NT (as global and local groups)
  - IBM’s OS/400
  - Oracle 8 onwards
  - .NET framework
- There is no generally accepted standard for RBAC
  - Role hierarchies
  - Semantics of role hierarchies



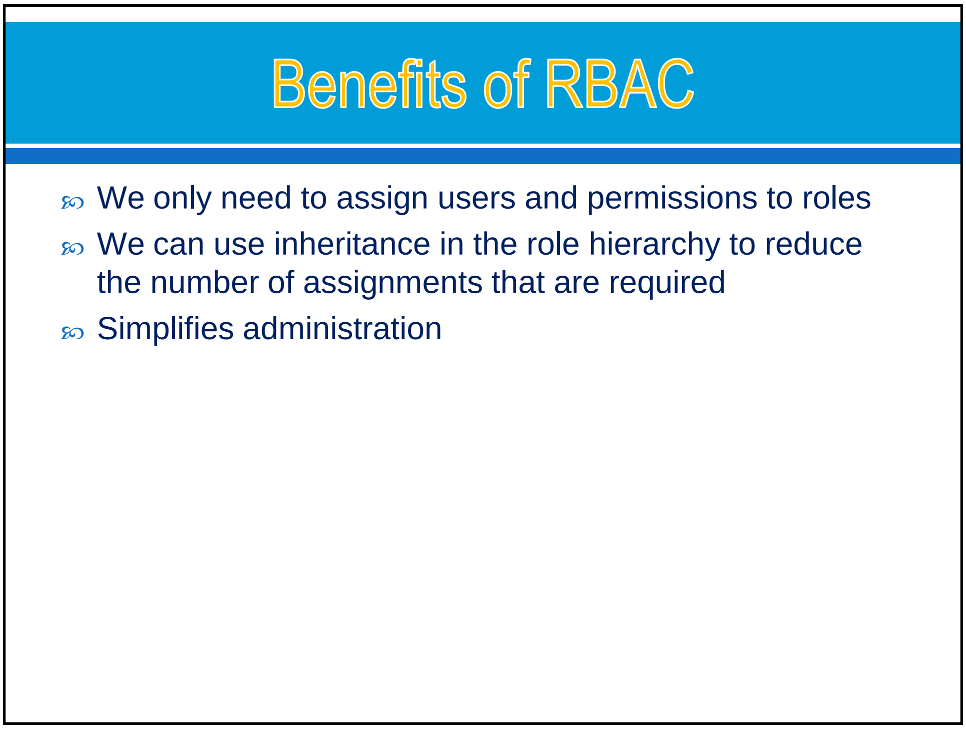
Need for Aggregate Models (RBAC)

- Practical ease of specification
  - Abstraction for users, permissions, constraints, administration
- Natural access control aggregations – based on organizational roles
  - As new employees join, their permission assignments are determined by their job
  - Permission assignment is largely static
- Central control and maintenance of access rights
- Flexible enough to enforce
  - least privilege, separation of duties, etc.



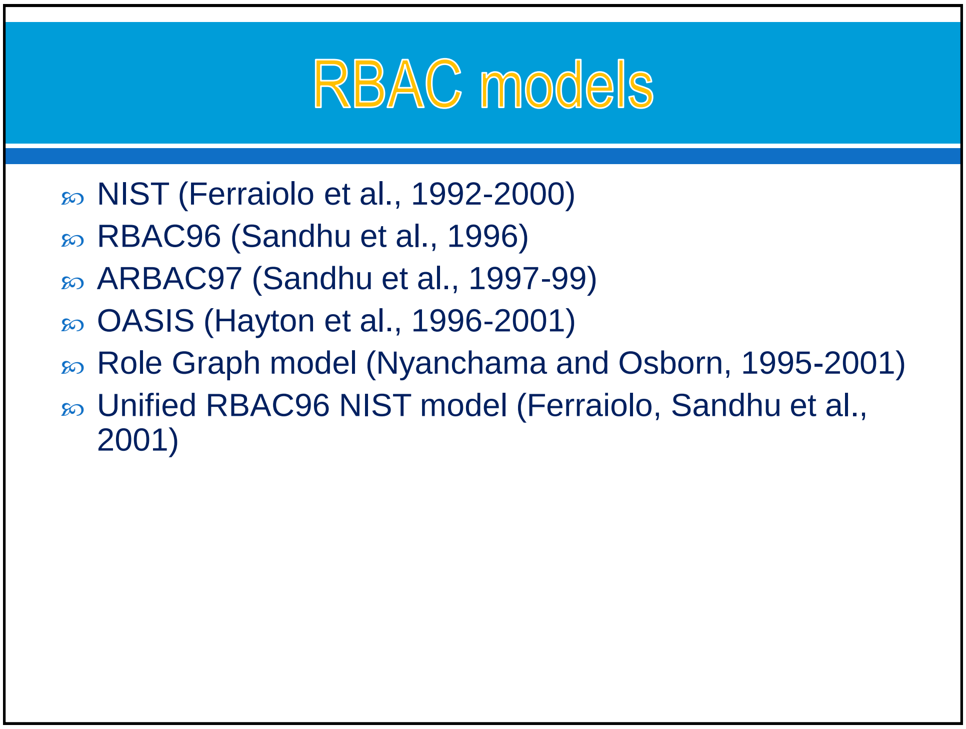
Does RBAC Achieve Its Goals?

- Practical ease of specification
  - Clear base model – need more help for constraints, admin
- Natural access control aggregations – based on organizational roles
  - In some cases, but not clear that organizational roles help with permission assignment – particularly with inheritance
- Central control and maintenance of access rights
  - Central view is a selling feature of products, but a single view of all can be complex (layering?)
- Flexible enough to enforce
  - Flexible access control expression, but difficult to determine if we enforce our security goals (constraints)



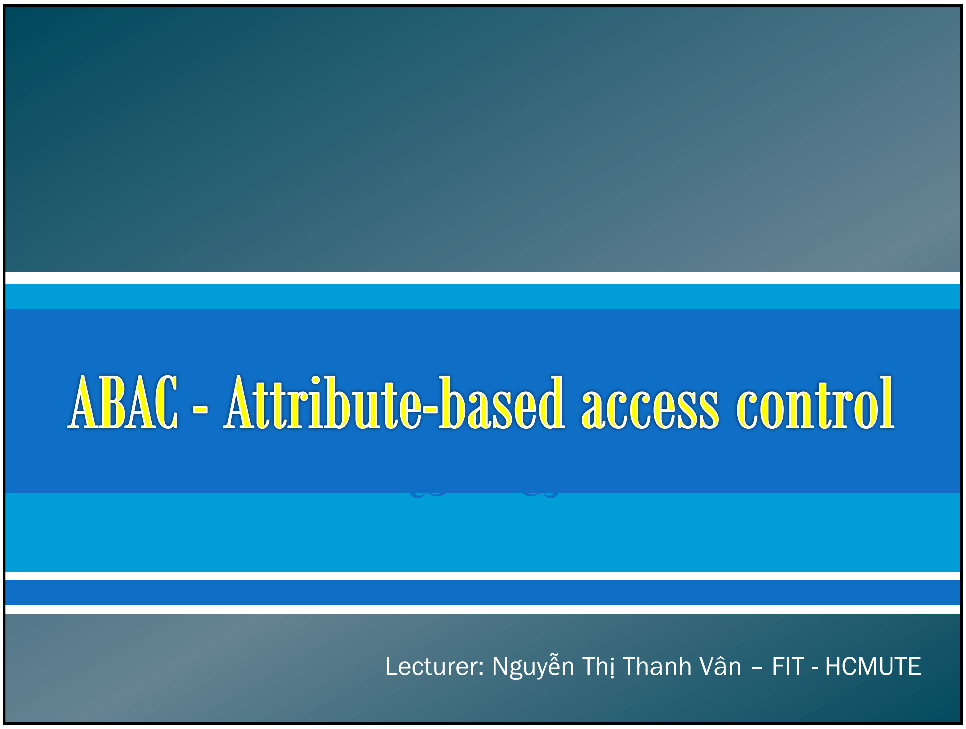
Benefits of RBAC

- We only need to assign users and permissions to roles
- We can use inheritance in the role hierarchy to reduce the number of assignments that are required
- Simplifies administration



RBAC models

- NIST (Ferraiolo et al., 1992-2000)
- RBAC96 (Sandhu et al., 1996)
- ARBAC97 (Sandhu et al., 1997-99)
- OASIS (Hayton et al., 1996-2001)
- Role Graph model (Nyanchama and Osborn, 1995-2001)
- Unified RBAC96 NIST model (Ferraiolo, Sandhu et al., 2001)



ABAC - Attribute-based access control

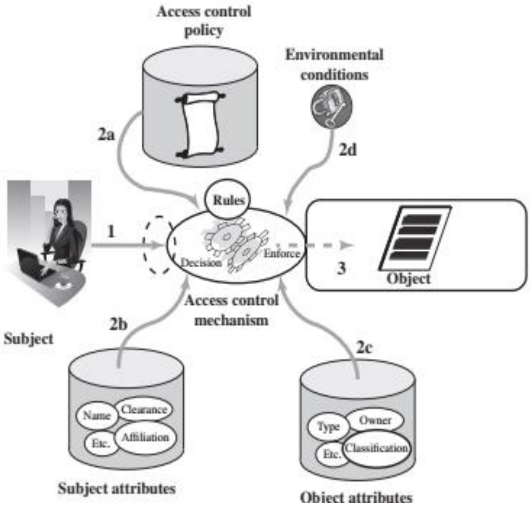
Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Attribute-based access control (ABAC)

- There are three key elements to an ABAC model:
  - attributes, which are defined for entities in a configuration;
  - a policy model, which defines the ABAC policies;
  - the architecture model, which applies to policies that enforce access control.

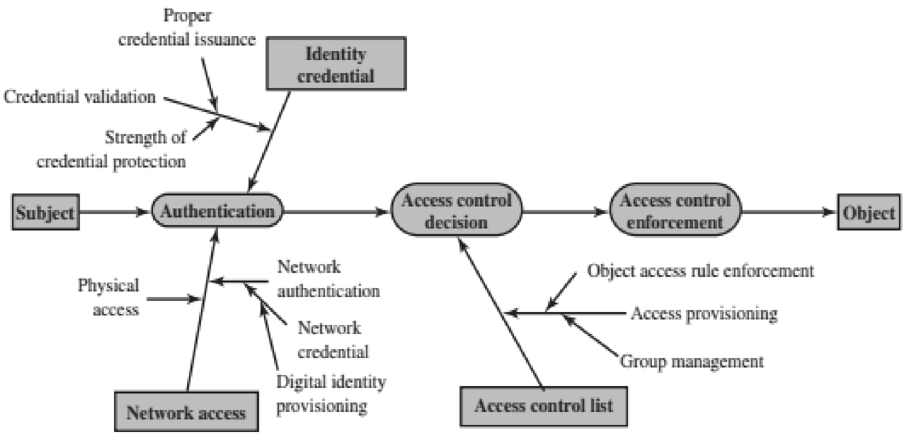
ABAC Logical Architecture

- An access by a subject to an object proceeds according to the following steps:
  - 1. A subject requests access to an object. This request is routed to an access control mechanism.
  - 2. The AC mechanism is governed by a set of rules:
    - (2a) that are defined by a preconfigured access control policy. Based on these rules, the AC mechanism assesses the attributes of the subject (2b), object (2c), and current environmental conditions (2d) to determine authorization.



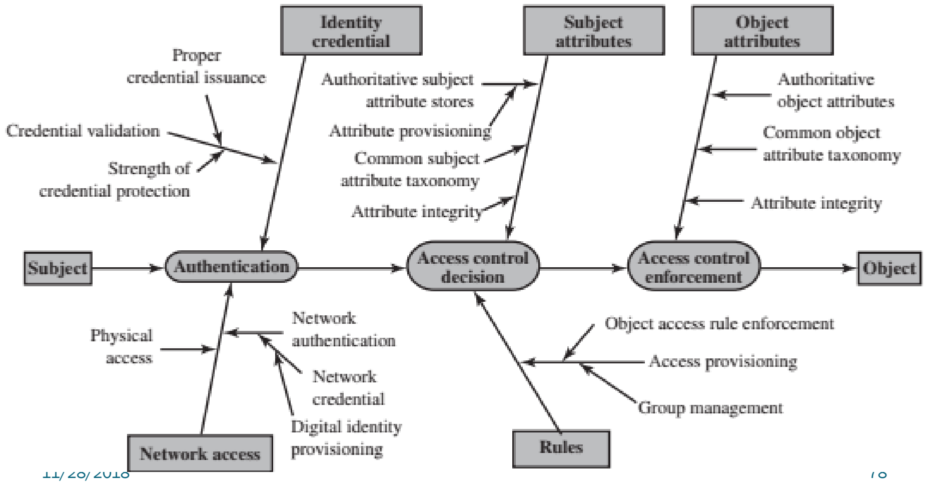
ACL and ABAC Trust Relationships

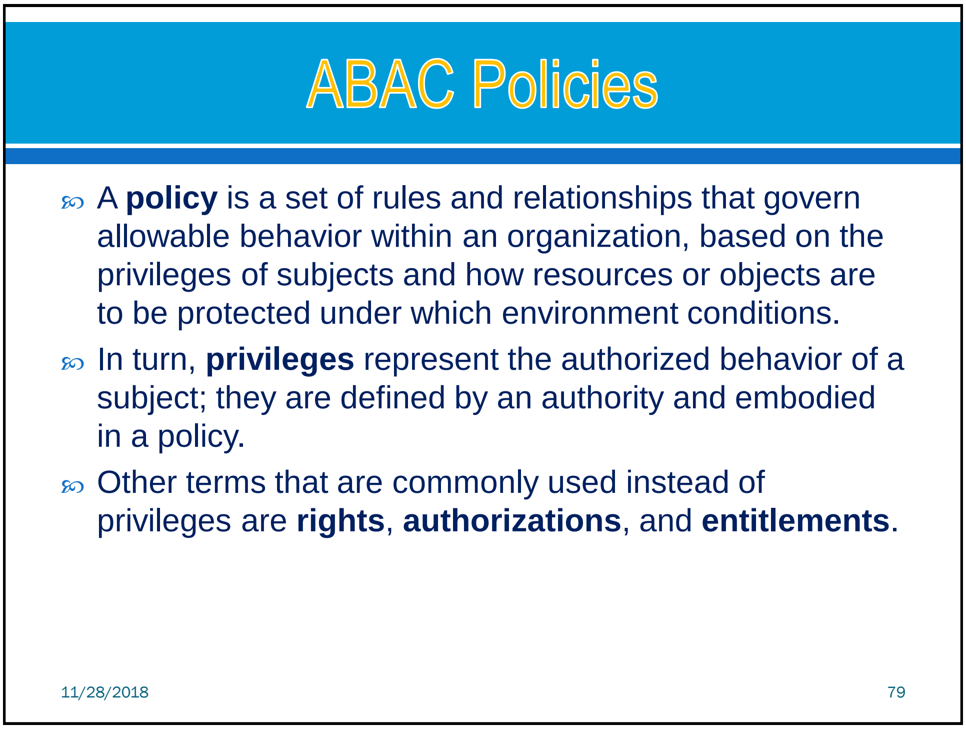
- ACL Trust Chain



ACL and ABAC Trust Relationships

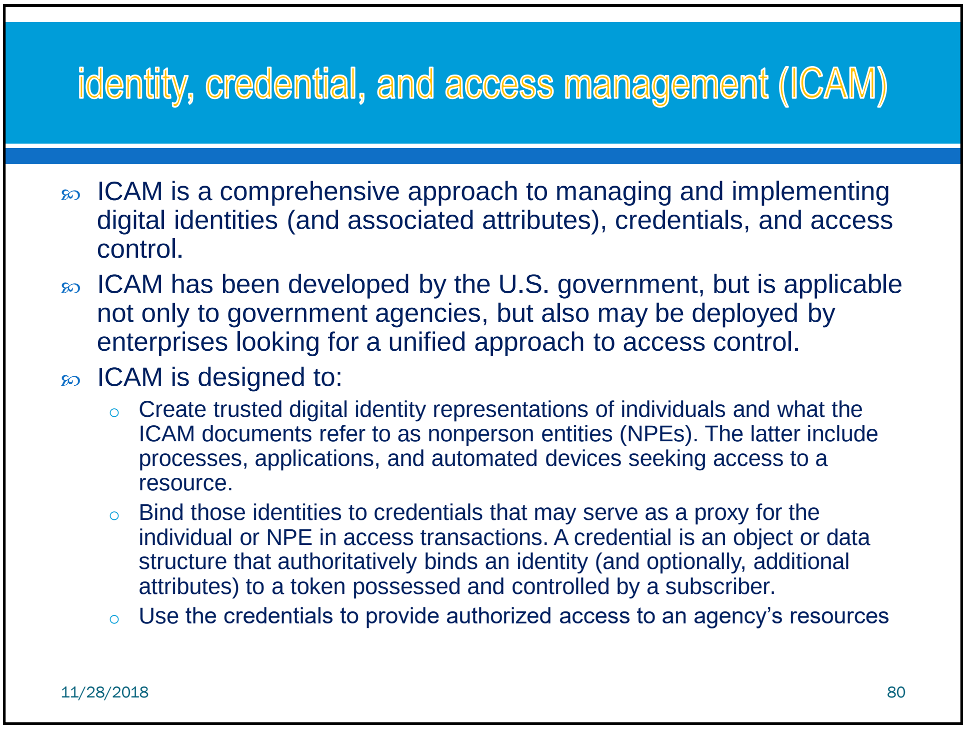
- ABAC Trust Chain





ABAC Policies

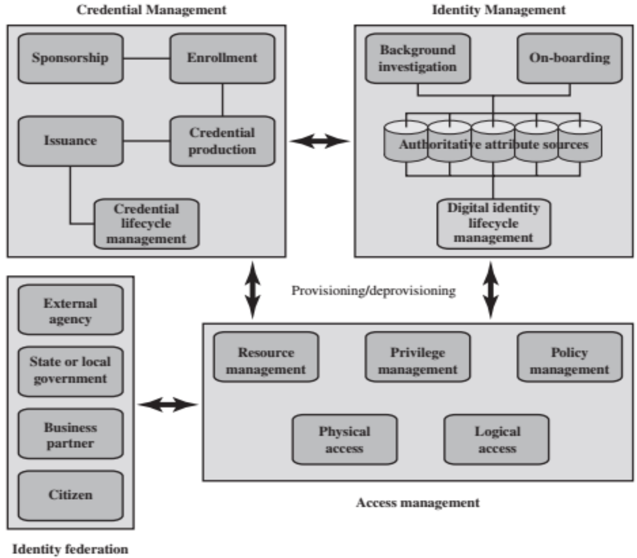
- A **policy** is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions.
- In turn, **privileges** represent the authorized behavior of a subject; they are defined by an authority and embodied in a policy.
- Other terms that are commonly used instead of privileges are **rights**, **authorizations**, and **entitlements**.



identity, credential, and access management (ICAM)

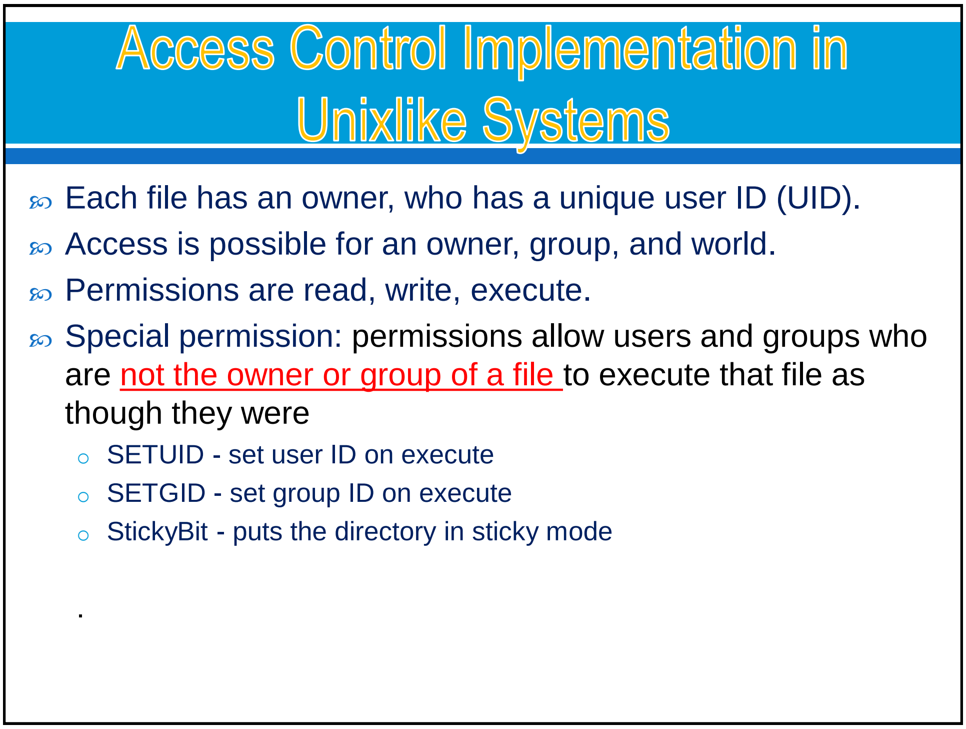
- ICAM is a comprehensive approach to managing and implementing digital identities (and associated attributes), credentials, and access control.
- ICAM has been developed by the U.S. government, but is applicable not only to government agencies, but also may be deployed by enterprises looking for a unified approach to access control.
- ICAM is designed to:
  - Create trusted digital identity representations of individuals and what the ICAM documents refer to as nonperson entities (NPEs). The latter include processes, applications, and automated devices seeking access to a resource.
  - Bind those identities to credentials that may serve as a proxy for the individual or NPE in access transactions. A credential is an object or data structure that authoritatively binds an identity (and optionally, additional attributes) to a token possessed and controlled by a subscriber.
  - Use the credentials to provide authorized access to an agency's resources

An overview of the logical components of an ICAM architecture



ICAM

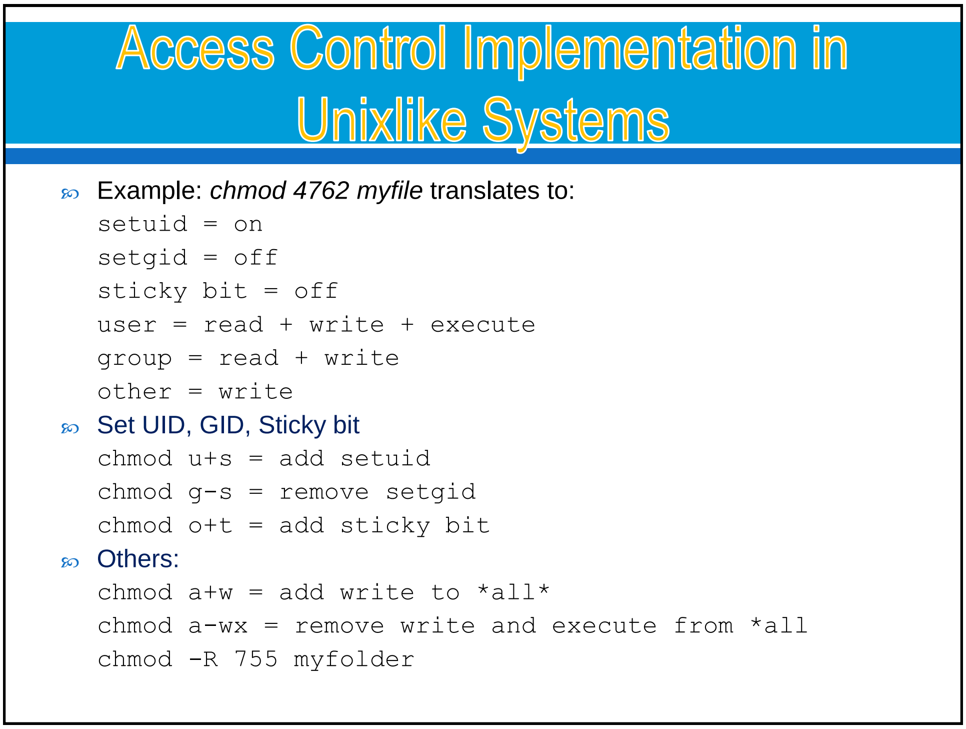
- CM is the management of the life cycle of the credential - an object or data structure that authoritatively binds identity (and optionally, additional attributes) to a token possessed and controlled by a subscriber.
- IM: is concerned with assigning attributes to a digital identity and connecting that digital identity to an individual or NPE. The goal is to establish a trustworthy digital identity that is independent of a specific application or context
- AM: the management and control of the ways entities are granted access to resources. It covers both logical and physical access, and may be internal to a system or an external element. It is used to ensure that the proper identity verification is made when an individual attempts to access security sensitive buildings, computer systems, or data.
- IF is a term used to describe the technology, standards, policies, and processes that allow an organization to trust digital identities, identity attributes, and credentials created and issued by another organization.



Access Control Implementation in Unixlike Systems

- Each file has an owner, who has a unique user ID (UID).
- Access is possible for an owner, group, and world.
- Permissions are read, write, execute.
- Special permission: permissions allow users and groups who are not the owner or group of a file to execute that file as though they were
  - SETUID - set user ID on execute
  - SETGID - set group ID on execute
  - StickyBit - puts the directory in sticky mode

.



Access Control Implementation in Unixlike Systems

- Example: *chmod 4762 myfile* translates to:

```
setuid = on
setgid = off
sticky bit = off
user = read + write + execute
group = read + write
other = write
```

- Set UID, GID, Sticky bit

```
chmod u+s = add setuid
chmod g-s = remove setgid
chmod o+t = add sticky bit
```

- Others:

```
chmod a+w = add write to *all*
chmod a-wx = remove write and execute from *all
chmod -R 755 myfolder
```

ACL in Linux

- provide a finer-grained control over which users can access specific directories and files.
- Using ACLs, you can specify the ways in which each of several users and groups can access a directory or file.
- Commands:
  - displays the file name, owner, group and the existing ACL for a file:

getfacl

  - sets ACLs of files and directories: **setfacl -m**

```
setfacl -m ugo:u/g_name:permissions fil/fol_name
```

  - removes rules in a file or folder's: **setfacl -x**

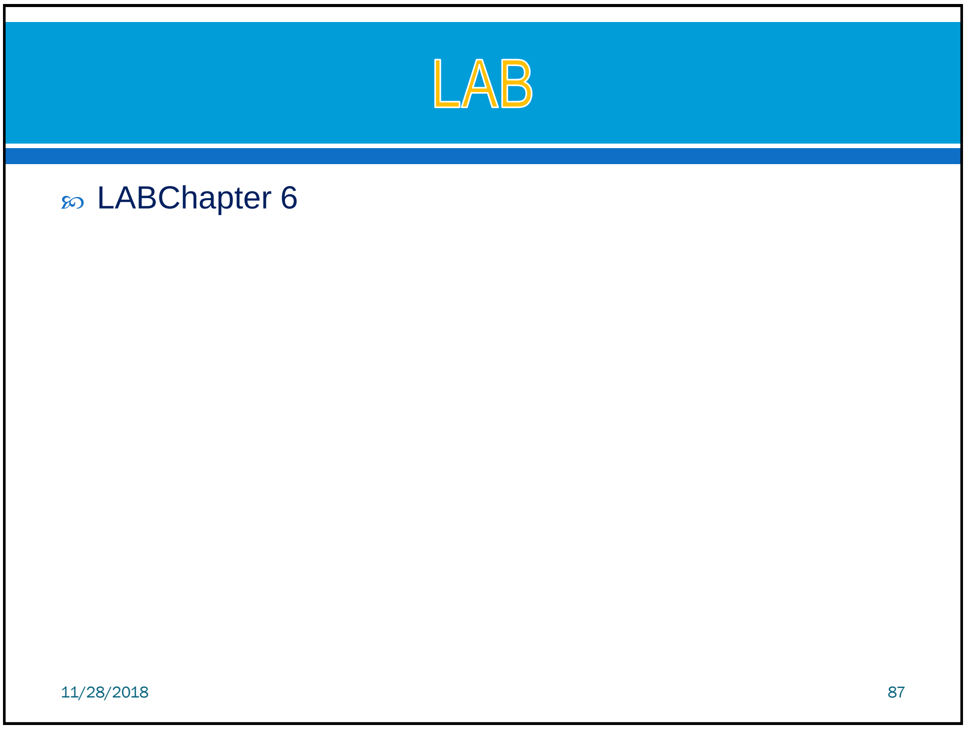
Use numeric or character to set permission

thêm cờ

ACL in Windows

- Commands:
  - List: **net user, net localgroup**
  - Change the permisions
  - Testing - quickly start a program as another user: **runas**

Ex, `runas /User:jack cmd.exe`



LAB

- LABChapter 6